jpsoftware

CMDebug 32

Table of Contents

1   Overview

jpsoftware

CMDebug 32

Welcome to our help! We have designed this help file to accompany our products *CMDebug* and *TCC-RT*.

CMDebug is a rich development environment that allows you to:

- Create and edit Windows batch files using a full featured editor, including syntax highlighting.
- Debug batch scripts with a sophisticated debugger, including single-stepping, conditional breakpoints, a watch window, and profiling.
- Choose between CMD.EXE compatibility or the much more powerful *TCC-RT* scripting language (a superset of CMD's scripting language).

CMDebug is designed for Windows 10, Windows 11, Server 2016, Server 2019, and Server 2022.

JP Software Inc.
web:https://jpsoft.com/

2 What's New in Version 32

CMDebug 32.10:

IDE / Batch Debugger:

Added support for creating & editing NTFS streams. The syntax is "*filename.ext:streamname*".

New Internal Variables:

_BATCHLABEL - Returns the name of the current GOSUB subroutine. (Or an empty string if not in a subroutine.)

Updated Commands:

COPY

/G - now writes the progress % directly to the display instead of STDOUT.

DO / ITERATE

ITERATE *n* - if *n* is specified and > 0, ITERATE will exit *n* nested DO loops and then iterate the *n*th parent DO loop.

MOVE

/G - Now writes the progress % directly to the display instead of STDOUT.

POPD

/N - Pop the directory off the stack, but don't change the directory.

PUSHD

/N - Push the directory onto the stack, but don't change the directory.

RD

/Nj - Don't follow junctions / symlinks

WATCH

The WATCH header now displays a timestamp before the command name.

/A - Highlights all changes between the current run and the first one,instead of the difference between the current and previous runs.

/B - Beeps if the return code != 0

/F - Freezes the display if hte output changed, and prompts you to enter a key to continue.
.
/U - Beep if the output changes

/X - Exits if the output changes

New Commands:

CALLER

Returns the context of the current batch call, including the line number, (optional) subroutine label, and batch file or library name. The syntax is:

CALLER *n*

If *n* is not specified, CALLER displays the line number and batch file or library name for the current line. If *n* is specified, CALLER displays the line number, subroutine label (or "main" if not in a subroutine), and batch or library name. The *n* value for the current line is *0,* the previous call in the program stack is *1, etc.*

EXEC

Replace the current TCC shell with the specified program. The syntax is:

EXEC *command [args ...]*

FALSE

Returns 0 (and sets ERRORLEVEL to 0).

FSEARCH

Search files for text. (FSEARCH is the replacement for the aged and soon-to-be-deprecated FFIND, and the options and syntax are almost identical. New features will only be implemented in FSEARCH, not FFIND. (FSEARCH is entirely new code; it doesn't share anything with FFIND.) The FSEARCH syntax is:

FSEARCH [/= /+*n* /-*n* /8 /B /C /E"*regex*" /F /G /H /I /L /N[dehjs] /Q /S[[+]*n*] /T"*text*" /U /V /Y /Z] [*path*] *filename*

   /+*n* - Skip the first *n* matches

   /-*n* - Stop after *n* matches

   /8 - Instead of scanning the files for their type, they are assumed to be UTF8 (this is a little faster).

   /B - Only display filenames (no header or footer or summary or matching lines)

   /C - Match case

   /E"..." - Regular expression search

   /F - Stop after first match (overrides /V)

   /G - Change to the directory containing the first matching file (also sets /F and overrides /V)

   /H - Don't search for text in binary files. By default, this includes
.exe, .dll, .sys, .chm, .zip, .pdb, .pch, .obj,
.tar, .com, and .ewriter. You can define your own list by setting the "BINARY_FILES" environment variable.

      For example, to ignore .exe, .sys, and .dll files:

         BINARY_FILES=.exe;.sys;.dll

   /I - Used with /T to tell FSEARCH to ignore wildcard characters (*, ?, and [...]).

   /L - Display line numbers for matching text

   /N... - Disable options:

      D - Don't scan hidden subdirectories

E - Don't display errors

H - No header

J - Skip junctions

S - No footer (summary)

/Q - Don't display any output. The internal variables (see below - _fsearch_errors, _fsearch_files, and _fsearch_matches **are** set).

/S - Search subdirectories of the specified (or default) path.

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.

If you specify a + followed by a number after the /S, FFIND will not search for files until it gets to that depth in the subdirectory tree. For example, if you have a directory tree \a\b\c\d\e, /S+2 will not find anything in \a or \a\b.

/T"..." - Search for the matching text. Supports TCC wildcards (?, *, and [...]).

/U - Only display summary line (no filenames or matching lines; overrides /V)

/V - Display all matching text (FFIND defaults to only displaying the first match in each file)

/Y - Display a "Continue Y/N" prompt after displaying each match

/Z - Highlight the matched text

FSEARCH will automatically determine the file type (ASCII, UTF8, or UTF16).

FSEARCH sets three internal variables:

_fsearch_errors - Errors when running FSEARCH (i.e., file/path not found, file locked, access denied, etc.)

_fsearch_files - The number of files containing one or more matches

_fsearch_matches - The total number of matches

INTERNAL

Runs the specified internal command, ignoring any aliases, plugins, or external apps. The syntax is:

INTERNAL *command args ...*

If *command* is not an internal TCC command, INTERNAL will return an error.

RANDOM

Generates a random integer, bool, float, char, or string. The syntax is:

RANDOM /I *start, end* - Create an integer >= *start* and <= *end*. The *start* and *end* arguments are signed 64-bit values.

RANDOM /B - Create a boolean (0 or 1)

RANDOM /F *start, end, precision* - Create a floating point number >= *start* and <= *end,* with *precision* decimal places.

RANDOM /C *start, end* - Create a character between (inclusive) the characters *start* and *end*.

RANDOM /S *start, end, length* Create a string composed of characters between *start* and *end* (inclusive).

RANDOM /D *n* - Roll an *n* sided dice.

[TRUE]

Returns 1 (and sets ERRORLEVEL to 1).

CMDebug 32.0:

CMDebug and *TCC-RT* are built with a new version of Visual Studio.

TCC-RT:

Many security, performance & size improvements.

Added support for 8K monitors.

Added support for Python 3.13.

IDE / Batch Debugger:

Many security, performance & size improvements.

Added support for 8K monitors.

Improved the syntax styling performance for large files.

The editor now supports autocompletion for *TCC* or CMD command names, internal variables, and variable functions. To display the autocompletion dropdown, enter the partial name and then press Ctrl-Enter.

There are two new options in the File menu:

Save to HTML - saves the current file as an HTML file.

Save to XML - saves the current file as an XML file.

There is a new option in the View menu:

Mark max colum - Invokes a dialog box asking for a column #. Then draws a vertical line following that column in the edit window. This can be useful to identify overly-long lines.

Help:

Help for CMDebug is now available in English, French, German, and Spanish. You can download the help files from:

[JP Software Downloads - Take Command, TCC, CMDebug, and TCC-RT](#)

The eWriter file viewer has been updated to version 4.0, and the skin for the Take Command help has been rewritten.

The help is build with a new version of Help & Manual (9.4.1).

New Commands:

[WATCH](#)

Run command(s) repeatedly, displaying the output and highlighting the difference(s) from the last run. WATCH allows you to see how program output changes over time.

The syntax is:

WATCH [/C /D /H*n* /I*n* /Mn */Nf /Nh* /R"regex" /T*n* /V] "*command ...*"

/C	Clear the display and home the cursor before each run
/D	Disable the highlight colorization
/H*n*	Display only the leading *n* lines
/I*n*	Interval (in seconds) between each run
/M*n*	Maximum number of times to run the command
/Nf	Disable the WATCH footer
/Nh	Disable the WATCH header
/R"regex"	Only display the output lines that match the regular expression
/T*n*	Display only the trailing *n* lines
/V	Verbose output (header and footer)
Command	Command(s) to execute

Command can be an internal command, alias, batch file, or an external application.

A simple example to display how WATCH works:

    watch /c /v "echo time=%%_time & echo date=%%_date"

Note that you need to double your %'s if you want the variables to be expanded by the specified commands instead of by WATCH.

2.1 What's New in Version 31

CMDebug:

Many security, performance & size improvements.

The Scintilla edit control has been updated to version 5.3.7.

The Lexilla syntax coloring control has been updated to version 5.2.7.

The Toolbox window has been updated with all of the new commands, command dialogs, variables, and functions since v27.

The "Go To" dialog now shows the current line, the current column, and the total number of lines in the tab window.

There is a new "RegEx Filter" combobox on the toolbar. If you enter a string in the filter box, the debugger will hide all non-matching lines in the current tab window.

There are three new features in the "Find" dialog box:

- The Find text edit control is now a combobox, so you can retrieve prior search strings
- There is a new "Wrap" checkbox option that will (continuously) loop through the document
- There is a new "Mark All" button that will place a bookmark on each matching line in the document. Marking supports either extended TCC wildcards (i.e., *, ?, []) or regular expressions. If you enter a string with no wildcards, a * will be prefixed and appended to the string (i.e., *string*) to find a match anywhere in the line.

There are five new features in the "Replace" dialog box:

- The Find text edit control is now a combobox, so you can retrieve prior search strings
- The Replace text edit control is now a combobox, so you can retrieve prior replacement strings
- There is a new "Wrap" checkbox option that will (continuously) loop through the document
- The regular expression option supports RE replacements
- The Replace dialog now displays the number of replacements in the bottom right

TCC-RT:

Many security, performance & size improvements.

TCC has a new pseudo-character device. TMP0: - TMP9: are similar to CLIP-: - CLIP9:, but are a little faster because they always work in UTF16 (so they don't translate to/from ANSI), and they don't need to access the Windows Clipboard (for CLIP0:). They also do not rotate like CLIPn: when

something is pasted to the Windows Clipboard. Like CLIP*n:*, TMP*n:* values are local to the current session of TCC-RT.

Prefixing a command name with a * will now ignore library function name matches (as well as the previous behavior of ignoring alias name matches).

Help:

The eWriter file viewer has been updated to version 4.0, and the skin for the *CMDebug* help has been rewritten.

The help is build with a new version of Help & Manual (9.2).

New Internal Variables:

_PBATCHNAME - Returns the name of the parent batch file. If there is no parent (the batch file was not CALL'd), it returns an empty string.

Updated Internal Variables:

_VOLUME - Added workaround for Windows 10/11 bug on some systems.

New Variable Functions:

@TMP[n,id] - Returns line *n* from the specified TMP device (0 - 9).

@TMPWN[id, string] - Writes the *string* to the specified TMP device (0 - 9).

Updated Variable Functions:

@WSLPATH - Now does a "truename" to see through network assignments, junctions, symbolic links, SUBST's, and home directory references.

Updated Commands:

CLIP

The /C and /S options accept the clipboard argument either as a digit (0 - 9) or a string (CLIP0: - CLIP9:).

COPY

/DS:[acwu]yyyy-mm-dd - Change the date timestamp on the target file(s) to the specified date.

/RCT - Request the transfer channel compress the data during the copy operation. In Windows 10+, this option is supported for files residing on SMB shares where the SMB protocol version is v3.1.1.1 or greater.

/TS[acwu]hh:mm:ss.ms - Change the time timestamp on the target file(s) to the specified time.

DO

/Y - Read a one-dimensional array and assign each value to the DO variable. For example:

```
do x in /Y MyArray
    echo x = %x
enddo
```

INKEY

/T - display a countdown timer (/Wn is also required).

INPUT

/T - display a countdown timer (/Wn is also required).

LIBRARY

Prefixing a command name with a * will now ignore library function name matches (as well as the previous behavior of ignoring alias name matches).

The LIBRARY command dialog now supports the /N (display library name + function) and /Q (don't display errors) options.

MOVE

/DS:[acwu]yyyy-mm-dd - Change the date timestamp on the target file(s) to the specified date.

/TS[acwu]hh:mm:ss.ms - Change the time timestamp on the target file(s) to the specified time.

REBOOT

/A - Restart applications. Shuts down the system and then restarts it, as well as any applications that have been registered for restart using the Windows RegisterApplicationRestart API.

/F - Force the specified reboot option. This option does not send the WM_QUERYENDSESSION message to applications, so this can cause applications to lose data. This option is only valid when used with the /P, /R, or /S options.

/H (hybrid shutdown) - Prepare the system for a faster reboot. This option is only valid wihen used with the /P, /R, or /S options.

RESOLUTION

Now displays the scaling factor (100% - 500%) and the DPI (x and y) for each monitor.

TASKLIST

/Nf - don't display the TASKLIST footer.

New Commands:

TMP

TMP displays or modifies the 10 TMP pseudo-character devices available in *TCC* (TMP0: - TMP9:). The syntax is:

TMP [/C *clipn:* /S *clipn: text*]

/C - Clears TMP device *n*

/S - Sets TMP device *n* to *text*

If you don't specify any arguments, TMP will display the current contents of TMP0: - TMP9:.

2.2   What's New in Version 30

<u>**CMDebug:**</u>

The *CMDebug*, and *TCC-RT* installers are built with a new version of Advanced Installer.

The installer will not display the "Thank You" page when installation is complete if a silent install was requested (requires an elevated session).

Many security, performance & size improvements.

The GUI framework library has been updated.

The Scintilla edit control has been updated to version 5.3.4.

The Lexilla syntax coloring control has been updated to version 5.2.4.

Add 6 new themes:

VS2019
VS2019 Dark
VS2019 Blue
VS2022
VS2022 Dark
VS2022 Blue

/Wrap:*n* - Startup option to set the default line wrapping mode. *n* is a value from 0 - 3.

0   None
1   Word
2   Character
3   Whitespace

There is a new submenu in the View menu to set the line wrapping mode. Line wrapping mode is unique to each tab window, so you can have some tabs wrapping and others not.

Wrap
    None
    Word
    Character
    Whitespace

There is a new entry in the Edit menu - "Duplicate" will copy the existing line and insert it above the current line.

The Tools menu has a new option to invoke the new search & replace command (see SREPLACE below).

TCC-RT:

Many security, performance & size improvements.

Added Python 3.12 support.

The Lua interpreter has been updated to version 5.4.6.

There is a new piping option - **|&|** means "only pipe STDERR".

TCC now supports **-** (hyphen) as a shorter version of **CON:**.

Help:

The help is built with a new version (9.0.3) of Help & Manual.

The eWriter file viewer has been updated to version 3.4.0, and the skin for the *CMDebug* help has been rewritten.

New Internal Variables:

_IPV6 - Returns the IPv6 address of the local computer. If the computer has more than one NIC, _IPv6 returns a space-delimited list of all IPv6 addresses.

New Variable Functions:

@JSONNODENAMES["filename",]xpath] - Returns a space delimited list of the element names for the specified xpath.

@PINGR[host[,timeout[,packetsize[,ttl[,type]]]]] - Returns the address of the host responding to the PING (ICMP ECHO) request. This may or may not be the host specified in the first argument.

@XMLNODENAMES["filename",]xpath] - Returns a space delimited list of the element names for the specified xpath.

Updated Variable Functions:

@EVAL - Added support for octal numbers. Prefix them with a "0o" (zero + lower case 'o'):

@eval[0o10+0o11]

You can also specify the output as octal:

@eval[123+456=o]

@SERVICE[service,info] - There are two new options for *info*:

6 - Return the process ID for the service

7 - Return the service flag. The flag can be one of the following values:

0 - The service is running in a process that is not a system process, or is not running.

1 - The service is running in a system process that must always be running

@WINAPI - added a new argument type "PINT64=n" for 64-bit integers.

Updated Commands:

IFTP

/O=0|1 - Specifies whether or not IFTP should overwrite downloaded files. If /O=1, an error will be thrown whenever the local file exists before a download operation.

/SSL=0|1 - Specifies whether TLS/SSL is enabled in IFTP. When 0 (the default) the class operates in plaintext mode. When 1, TLS/SSL is enabled.

SENDMAIL

SENDHTML has some new options:

/Command="*command*" - Send additional commands directly to the server. You can specify multiple /Command="..." arguments.

/Date="*date*" - Create a Date SMTP header and attach it to the message. (If this option is not set, the default SENDHTML behavior is to create a Date SMTP header reflecting the current date and time when the message is sent.) RFC 822 contains detailed date format specifications. An example of a valid date is "Mon, 1 May 20:15:00 EST".

SENDHTML

SENDHTML has some new options:

/Command="*command*" - Send additional commands directly to the server. You can specify multiple /Command="..." arguments.

/Date="*date*" - Create a Date SMTP header and attach it to the message. (If this option is not set, the default SENDHTML behavior is to create a Date SMTP header reflecting the current date and time when the message is sent.) RFC 822 contains detailed date format specifications. An example of a valid date is "Mon, 1 May 20:15:00 EST".

/Image="*imagefile*" - Embed an image in the HTML message. You can specify multiple /Image="..." arguments.

SETARRAY

You can now combine the /R and /F options.

SMPP

/Priority=*n* - This option tells the server what type of priority to assign to the message. The possible values are:

0 - Low
1 - Normal
2 - High
3 - Urgent

The effect of the message priority setting is dependent upon the Message Center manufacturer and the network on which the target recipient lies. For example, some MCs may immediately forward "urgent" messages, some networks may use the priority setting as a visual indicator of the message's urgency (e.g. blinking icons, etc.), and some networks may entirely ignore the priority setting.

/SMPPVersion=*n* - The SMPP version to be used throughout the connection.

0 - 5.0
1 - 3.4
2 - 3.3

The default value is version 3.4 as it is the most widely used version. If version 5.0 is supported it is recommended.

/SSLMode=*n* - TheDetermines how SMPP starts SSL negotiation

0 - Automatic
1 - Implicit. The SSL negotiation will start immediately after the connection is established.
2 - Explicit. SMPP wil first connect in plaintext, and then explicitly start SSL negoatiation through a protocol command such as STARTTLS.
3 - None. No SSL negotiation, no SSL security. All communication will be in plaintext mode.

The default value is 3.

START

/Env="*filename*" - Creates a new environment for the process using the contents of *filename.* The format of *filename* is:

var1=value1
var2=value2

...

Because some Windows API calls will fail if there is no "SystemRoot" variable in the environment, **TCC** will add the existing *SystemRoot* value to the new environment if wasn't specified in *filename*.

/Idle=*n* - Waits until the started process to finish processing its initial input and is waiting for user input, or until the timeout period has elapsed. *n* specifies the timeout period in milliseconds.

TASKEND

TASKEND now supports terminating processes on remote systems. The syntax is:

TASKEND /Remote="*remotename*" /User="*username*" /Password="*password*" *PID*

You must have have debug privileges on the remote system.

WHICH

If the command is a symbolic link and you used the /A option, WHICH will display the symbolic link for the executable.

New Commands:

DIRENV

Configures the environment on a per-directory basis.

The syntax is:

DIRENV [ON | OFF]

When DIRENV is on, TCC will look for a file called ".envtc" when it changes directories. The format of .envtc is:

var1=value1
var2=value2
...

When TCC enters a directory, it will look in .envtc for environment variables to set while in that directory. When TCC leaves a directory, it will unset the variables in .envtc.

EXPR

A greatly enhanced version of the Linux expression evaluator. The syntax is:

EXPR *expression*

Expression can be:

string : regex	Returns position of regular expression *regex* in *string*
match *string regex*	Same as *string : regex*
substr *string pos len*	Substring of *string*, *pos* starts at 1
index *string chars*	Index in *string* (first character is 1) where anything in *chars* is found, or 0 if nothing matches
length *string*	Length of *string*
arg1 [operator] arg2...	This can be any arithmetic expression supported by @EVAL, or any conditional expression supported by IF /IFF.

If you have special characters (i.e., < > & |) on the line you must either enclose the entire expression in double quotes (EXPR will remove them before evaluating the expression) or escape them.

The regular expression match is always anchored (there is an implied leading ^). If the regular expression contains (**…**), and it matches at least part of *string*, **EXPR** returns that part of *string*; if there is no match, **EXPR** results in 0. If the regular expression doesn't contain (..), the result is the number of characters matched. MATCH performs the same operation as the colon operator.

Comparisons are arithmetic if both ARGs are numbers, otherwise the comparisons are string-based.

WAITFOR

Waits for an app to exit, or optionally for the app to finish processing its initial input and wait for user input.

The syntax is:

WAITFOR [/Exit=*n* /Idle=*n*] [PID | *title* | *exename*]

/Exit - Wait for a maximum of *n* milliseconds for the process to exit. *n* will default to 10000ms (10 seconds).

/Idle - Wait for a maximum of *n* milliseconds for the process to enter the idle state. *n* will default to 10000ms (10 seconds).

PID - The process ID. This can be either hex or decimal; if it is hex you must prefix it with a "0x".

title - The Window title. The title must be enclosed in double quotes; wildcards are supported.

exename - The name of the executable file.

WEBSOCKET

Establish a WebSocket connection to a server and send a string. The syntax is:

WEBSOCKET [/V /Origin=*server* /User=*user* /Password=*password*]] *"ws:servername"* *string*

/V(erbose) - Display status messages

/Origin - If specified, WEBSOCKET will include an Origin HTTP header in the connection request with the value provided. Servers may use this value to validate requests. Servers may reject requests depending on the value provided. A typical value that would be set is of the form "http://example.com".

/User - The user name if authentication is used

/Password - The password if authentication is used

servername - The WebSocket server to connect to. For example: "ws://echo.websocket.org"

string - The text to send to the server

The options are position dependent; they can only appear at the beginning of the command line in the order specified above.

2.3   What's New in Version 29

<u>**CMDebug:**</u>

The *CMDebug*, and *TCC-RT* installers are built with a new version of Advanced Installer.

The installer will not display the "Thank You" page when installation is complete if a silent install was requested (requires an elevated session).

Many security, performance & size improvements.

The GUI framework library has been updated.

The Scintilla edit control has been updated to version 5.3.1.

The Lexilla syntax coloring control has been updated to version 5.2.0.

There are nine new themes:

    Visual Studio 2017
    Visual Studio 2017 Dark
    Visual Studio 2017 Blue
    Visual Studio 2019
    Visual Studio 2019 Dark
    Visual Studio 2019 Blue
    Visual Studio 2022
    Visual Studio 2022 Dark
    Visual Studio 2022 Blue

The *CMDebug* editor has improved support for very large files.

The *CMDebug* editor will now display document changes in the margin and in the text. In the text, inserted characters appear with colored underlines and points where characters were deleted are shown with small triangles. The margin shows a block indicating the overall state of the line. The states are modified (orange), saved (green), saved then reverted to modified (green-yellow), and saved then reverted to original (cyan). The change history can be toggled on or off with the "Options / Change History" menu entry.

The *CMDebug* editor now supports horizontal mouse wheel scrolling (Shift+wheel).

*/*BREAKPOINT:*n* - *CMDebug s*tartup option to set a breakpoint on the specified line in the file after opening the tab window.

<u>**TCC-RT:**</u>

Many security, performance & size improvements.

Added support for Python 3.11.

You can return the string result of a command with %{command}. This is the same as @EXECSTR[command] but a little easier to write. For example:

    dir %{echo foo}

will be translated to "dir foo".

Array variables can now return a range of values. The syntax is:

    arrayvar[x..y]

TCC-RT will return the values from *arrayvar[x]* to *arrayvar[y]* with a space between each value.

TCC-RT now supports multiple clipboards. They are numbered from CLIP0: - CLIP9:. You can still use CLIP: - it is equivalent to CLIP0:. Clipboards 1 - 9 are only accessible to **TCC** internal commands and variable functions. External applications will only be able to access CLIP: / CLIP0:. For example:

    dir *.btm > clip1:
    dir *.exe > clip3:

When an app saves something to the default clipboard (CLIP: or CLIP0:), **TCC** will rotate the existing clipboard entries before saving the new CLIP0. CLIP0: will become CLIP1:, CLIP1: becomes CLIP2:, etc. The old CLIP9: will be lost. If you save something to CLIP1: - CLIP9:, none of the other clipboard entries will be modified.

The **TCC-RT** specific clipboards (CLIP1: - CLIP9:) are always Unicode text.

See the new CLIP internal command for more details.

Help:

The help is built with a new version (8.5.0) of Help & Manual.

The eWriter file viewer has been updated to version 3.2, and the skin for the **CMDebug** help has been rewritten.

New Variable Functions:

@CLIPWN - Like @CLIPW, but accepts an optional clipboard number (0 - 9).

    @CLIPWN[*clipboard, line*]

@ODBCOPEN - Open a SQL database through the ODBC driver.

    @ODBCOPEN["*name*"]

@ODBCCLOSE - Close a SQL database through the ODBC driver.

    @ODBCCLOSE[]

@ODBCQUERY - Send a query to a SQL database through the ODBC driver. Returns the string result of the query. You must have called @ODBCOPEN or ODBC /O "*name*" before calling @ODBCQUERY.

    @ODBCQUERY[*arrayvar,* "*query*"]

arrayvar - An array variable that receives the output of the SQL query. (You must create it with SETARRAY before calling @ODBC.)

"query" - The SQL query to execute.

Updated Variable Functions:

@CLIP

    @CLIP has an optional second parameter (0-9) that specifies the clipboard you want to use (CLIP0: - CLIP9:). For example, to get the 5th line from CLIP7:

       @CLIP[5,7]

@FILEARRAY

   @FILEARRAY now supports clipboards 0 - 9.

@FILEDATE

   @FILEDATE now supports HTTP & HTTPS filenames, for last write only. Wildcards are not supported (HTTP limitation).

@FILETIME

   @FILETIME now supports HTTP & HTTPS filenames, for last write only. Wildcards are not supported (HTTP limitation).

@LINE

   @LINE now supports clipboards 0 - 9.

@LINES

   @LINES now supports clipboards 0 - 9.

@WINMETRICS

61	The default width, in pixels, of a maximized top-level window on the primary display monitor.
67	The value that specifies how the system is started: 0 Normal boot 1 Fail-safe boot 2 Fail-safe with network boot

@WINSYSTEM

120	121	The number of milliseconds a thread can go without dispatching a message before the system considers it unresponsive.
122	123	The number of milliseconds the system waits before terminating an application that does not respond to a shutdown request.
124	125	The number of milliseconds the service control manager waits before terminating a service that does not respond to a shutdown request.

Updated Commands:

COLOR

Updated /F to read improperly formatted .ITERMCOLORS files.

COPY

COPY now recognizes the (invalid) syntax "copy file1+,, file1" as a (dumb) way to fool CMD into TOUCH'ing the file with the current date. (The correct syntax is to leave off the trailing ",, file1". Or just use TOUCH.)

/BAK - If the target file exists, COPY will save it with a ".bak" extension before overwriting it. COPY will **not** create multiple versions of the .bak file; if you already have a *file.ext.bak*, it will be overwritten.

/DD - Remove any empty directories created with the /S option.

DIR

DIR now has limited support for HTTP & HTTPS filenames. DIR will display the filename, size, and date/time (for last write only). Wildcards are not supported (HTTP limitation).

LUA

LUA has been updated to version 5.4.4.

MOVE

/DD - Remove any empty directories created with the /S option.

START

/UNELEVATED - start the new process in an unelevated session. (Only necessary if TCC is running in an elevated session and you want to start a process unelevated.)

SYNC

/DD - Remove any empty directories.

WSHORTCUT

Added some new Windows folders:

> 3dObjectsFolder
> FrequentFolders

ReliabilityMonitor
RemoteAssistance
RemovableDrives
SystemRestore
TaskView
ThisDevice

New Commands:

CAPTURE

CAPTURE does video and / or audio screen capture. It supports H264, H265, VP80, VP90, MP3, FLAC, and AAC. The syntax is:

CAPTURE
"filename" [/Start=n /End=n /FPS=n /HWND=n /Monitor=n /Rect=top,left,bottom,right /Video =[H264 | HEVC | VP80 | VP90] /Audio=[MP3 | AAC | FLAC] /AudioFrom="name" /C /E /P]

    "filename" - The output filename (.mp4 or .asf for video; .mp3, .aac, .flac for audio)
    /Start - The start time in seconds (default 0)
    /End - The end time in seconds
    /FPS - Frames per second (default 25)
    /HWND - The window to capture
    /Monitor - The monitor to capture (1 - n)
    /RECT - The window rectangle to capture
    /Threads - The number of threads for the video encoding (default 1, maximum 16)
    /Video - Video encoding format (H264, HEVC, VP80, or VP90)
    /AudioFormat - Audio encoding format (MP3, AAC, FLAC)
    /AudioFrom - The friendly name of the audio source. You can use wildcards in the name; for example: /AudioFrom="HD Audio*"
    /C - Capture the cursor
    /P - Pause the capture
    /E - End the capture

If you do not specify /End, CAPTURE will continue capturing the screen until you call it again with the /E option.

If you do not specify /HWND or /RECT, CAPTURE will capture the desktop.

CAPTURE runs in a separate thread, so it will not block the current **TCC / Take Command** window.

CLIP

CLIP displays or modifies the 10 clipboards available in *TCC* (CLIP0: - CLIP9:). The syntax is:

CLIP [/C *clipn:* /R *n* /S *clipn: text*]

    /C - Clears clipboard *n*

    /R - Rotates the clipboards to make clipboard *n* the default (i.e., CLIP: / CLIP0:).

    /S - Sets clipboard *n* to *text*

If you don't specify any arguments, CLIP will display the current contents of CLIP0: - CLIP9:.

ODBC

Query a database through an ODBC driver. The syntax is:

ODBC [/O "*connectionstring*"]["*Query*"][/C]

/O - Send the specified connection string to the ODBC driver. This opens a persistent ODBC session.

/C - Close the ODBC session.

"*query*" - Executes a SQL query

PRINTF

Display a formatted string using the C Printf format. The syntax is:

PRINTF "*format string*" args ...

The arguments following the format string will be inserted in the output string according to the format type in the format string. The arguments can be variable names, variable functions, or literal strings; i.e.:

PRINTF "%s %d %x" %var1 999 %hexvar

The *format type* syntax is:

%[flags][width][.precision][length]type

flags	description
–	Left-justify within the given field width; Right justification is the default (see *width* sub-specifier).
+	Prefix the result with a plus or minus sign (+ or -) even for positive numbers. By default, only negative numbers are preceded with a - sign.
0	Prefix the number with zeroes (0) instead of spaces when padding is specified (see *width* sub-specifier).

width	description
number	Minimum number of characters to be printed. If the value to be printed is less than this number, the result is padded with spaces.
*	The *width* is not specified in the *format* string, but as an additional integer argument preceding the argument to be formatted.

.precision	description
.number	For integer specifiers (d, i, o, u, x, X): *precision* is the minimum number of digits to be written. If the value to be written is less than *precision*, the result is padded with leading zeros. For f and g specifiers: The maximum number of significant digits to be printed.

.*	The *precision* is not specified in the *format* string, but as an additional integer value argument preceding the argument that has to be formatted.

Type	**Output**
d *or* i	Signed decimal integer
u	Unsigned decimal integer
x	Unsigned hexadecimal integer
X	Unsigned uppercase hexadecimal integer
f *or* g	Decimal floating point
c	Character
s	String
%	A % followed by another % will write a single %

If you prefix a type with an **L**, PRINTF will insert commas as thousands separators. For example:

PRINTF "%Ld" 123456789

will output:

123,456,789

REPEAT

A simpler way than DO or FOR to execute a counted loop. The syntax is:

REPEAT *n command ...*

where *n* is the number of times you want to repeat *command*.

REPEAT sets the internal command variable **_repeat** to the current loop counter (1 to *n*).

2.4   Whats New in Version 28

CMDebug 28.0:

The *CMDebug* and *TCC-RT* installers are built with a new version of Advanced Installer.

Many security, performance & size improvements.

CMDebug / IDE is now CET Shadow Stack compatible.

The GUI framework library has been updated.

Improved support for high resolution displays and multiple monitors.

The Scintilla edit control has been updated to version 5.1. The lexer (lexilla.dll) has been separated from the editor (scintilla.dll). There are a number of improvements in readability and speed for high resolution displays.

TCC-RT:

Many security, performance & size improvements.

TCC-RT is now CET Shadow Stack compatible.

TCC-RT file expansion now supports "~\" (home directory) syntax. If the filename is ~, or begins with a **~\ (or ~/)**, *TCC* will substitute to the user's home directory, as defined by the HOME environment variable. (If HOME doesn't exist, *TCC* will look for %HOMEDRIVE + HOMEPATH.) For example:

    dir ~\
    copy foo ~\foofolder

TCC-RT file expansion now supports the predefined Windows folders. The syntax is **:foldername** where *foldername* can be:

    AccountPictures
    AdminTools
    AppCaptures
    ApplicationShortcuts
    CameraRoll
    CDBurning
    CommonAdminTools
    CommonOEMLinks
    CommonPrograms
    CommonStartMenu
    CommonStartMenuPlaces
    CommonStartup
    CommonTemplates
    Contacts
    Cookies
    Desktop
    DeviceMetadataStore
    Documents
    DocumentsLibrary
    Downloads
    Favorites
    Fonts
    GameTasks
    History
    ImplicitAppShortcuts
    InternetCache
    Libraries
    Links
    LocalAppData
    LocalAppDataLow
    LocalDocuments
    LocalDownloads
    LocalizedResourcesDir
    LocalMusic
    LocalPictures
    LocalVideos
    Music
    MusicLibrary

Nethood
OneDrive
OriginalImages
PhotoAlbums
PicturesLibrary
Pictures
Playlists
PrintHood
Profile
ProgramData
ProgramFiles
ProgramFilesX64
ProgramFilesX86
ProgramFilesCommon
ProgramFilesCommonX64
ProgramFilesCommonX86
Programs
Public
PublicDesktop
PublicDocuments
PublicDownloads
PublicGameTasks
PublicLibraries
PublicMusic
PublicPictures
PublicRingtones
PublicUserTiles
PublicVideos
QuickLaunch
Recent
RecordedTVLibrary
ResourceDir
RetailDemo
Ringtones
RoamingAppData
RoamedTileImages
RoamingTiles
SampleMusic
SamplePictures
SamplePlayLists
SampleVideos
SavedGames
SavedPictures
SavedSearches
Screenshots
SearchHistory
SearchTemplates
SendTo
SidebarDefaultParts
SidebarParts
SkyDrive
SkyDriveCameraRoll
SkyDriveDocuments
SkyDrivePictures
StartMenu

Startup
System
SystemX86
Templates
UserPinned
UserProfiles
UserProgramFiles
UserProgramFilesCommon
Videos
VideosLibrary
Windows

These folder names can be used in any internal *TCC-RT* command that takes filenames. For example:

dir :downloads
copy picture.jpg :pictures\myfolder1\

Help:

The help is built with a new version (8.3) of Help & Manual.

The .chm help (obsolete, unsupported, and deprecated by Microsoft) has been changed to .ewriter format. An eWriter eBook is WebHelp stored in a single file, and the eWriter.exe app is used to display the help. The eWriter format combines the benefits of CHM and WebHelp, eliminating the disadvantages of both. It uses the Windows HTML rendering engine to display the content, with support for CSS3, HTML5, JavaScript and media. This will allow us (in future versions) to incorporate tutorials and videos in the help file.

The new eWriter format includes support for high-resolution displays (.chm does not).

Unlike .chm help files, The eWrite HELP can be opened from network drives on local networks.

Repeated calls to HELP will open in the same help window.

Updated Variable Functions:

@FILESIZE - Now supports returning the size of file streams. @FILESIZE now also supports retrieving file sizes for HTTP and HTTPS files. (Note that due to HTTP protocol limitations, you cannot use wildcards or scan subdirectories.)

Updated Commands:

ACTIVATE

FORCEMIN - Force the window to be minimized even if the thread that owns the window is not responding.

VDESKTOP=*id* - Move the window to another virtual desktop. *id* can be either a desktop number (1-n), the GUID for that desktop, or the desktop name. See VDESKTOP for more details.

ASSOCIATE

/V:* - Displays all of the shell verbs and their commands for the specified extension.

DEL

If you are deleting a stream, DEL will check for a symlink and delete the stream from the linked file. (Windows does not support deleting a symlink'd stream.)

DIR

/\ - Display directory names with a trailing \.

ENUMSERVERS

/Domain=*domain* - The NetBIOS name of the domain to enumerate. If you do not specify a domain, ENUMSERVERS uses the primary domain.

IFTP

/K="..." - The CA signed client public key used when authenticating (SSH only). When authenticating via public key authentication this setting may be set to the CA signed client's public key. This is useful when the server has been configured to trust client keys signed by a particular CA. For example:

/K="SignedSSHCert=ssh-rsa-cert-v01@openssh.com
AAAAB3NzaC1yc2EAAAADAQABAAAB..."

The algorithm such as `ssh-rsa-cert-v01@openssh.com` in the above string is used as part of the authentication process. To use a different algorithm simply change this value. For instance all of the following are acceptable with the same signed public key:

- ssh-rsa-cert-v01@openssh.com AAAAB3NzaC1yc2EAAAADAQABAAAB...
- rsa-sha2-256-cert-v01@openssh.com AAAAB3NzaC1yc2EAAAADAQABAAAB...
- rsa-sha2-512-cert-v01@openssh.com AAAAB3NzaC1yc2EAAAADAQABAAAB...

/T=*nnn* - If this is set, the socket's keep-alive option is enabled, and TCP keep-alive packets will be sent periodically to maintain the connection. *nnn* is the inactivity time in seconds before a TCP keep-alive packet is sent.

LIBRARY

You can now specify which library to use for a function name (allowing you to use the same function names in different libraries). To specify a particular library and function, use the syntax:

library$function

Where *library* is the library file name, and *function* the name of the function.

If you don't specify a library name, **TCC** will use the old format and use the first matching function name it finds in the library list.

/N - LIBRARY with no arguments will display the function names in the library list. If you specify /N and no other arguments, LIBRARY will show the library name + function name in the *library$function* format.

PDIR

/\ - Display directory names with a trailing \.

START

/VDESKTOP=*id* - Start the app on another virtual desktop. *id* can be either a desktop number (1-n), the GUID for that desktop, or the desktop name. See VDESKTOP for more details. Note that Windows doesn't have an API to actually start on another desktop, so *TCC* starts it on the local desktop and them immediately moves it -- so you'll see a flash when the window starts and then disappears.

TASKEND

The PID can be hex if it is prefixed with a leading 0x.

/Ne - Don't display errors.

/R - Delete the process tree (the specified process and all of its child processes).

TASKLIST

The PID can be hex if it is prefixed with a leading 0x.

/R - Show the process tree (the specified process and all of its child processes).

TPIPE

Textpipeengine64.dll has been updated to version 11.8.1.

/Simple has some new redaction filters which are designed to work inside restriction filters.

    89 Remove diacritics
    90 Mainframe dump
    91 Redact x-over text
    92 Redact x-over digits
    94 Redact x-over non-blanks
    95 Replace with blanks
    96 Redact with pseudo NHS
    97 Redact with pseudo SSN
    98 Redact with pseudo bank number

TREE

/\ - Display directory names with a trailing \.

WINDOW

VDESKTOP=*id* - Move the window to another virtual desktop. *id* can be either a desktop number (1-n), the GUID for that desktop, or the desktop name. See VDESKTOP for more details.

WMIQUERY

WMIQUERY now supports remote queries. The namespace argument for remote servers will look like "\\remote-server\root\cimv2" (substitute your server name for "remote-server").

/USER=*username* - The user name to use for remote queries.

/PASSWORD=*password* - The password to use for remote queries.

/L - Don't separate records with a CR/LF. (This is probably only useful when you are querying single-line records.)

WSETTINGS

New settings dialogs (some require your system / device to be configured to support the option):

    ActivityHistory
    AdvancedDisplay
    AppDiagnostics
    AppVolume
    Audio
    AutomaticFileDownloads
    Broadcasting
    Clipboard
    DeliveryOptimization
    Documents
    DownloadMaps
    Encryption
    EyeTracker
    FindMyDevice
    Fonts
    GameBar
    GameDVR
    GameMode
    GraphicsSettings
    InkingAndTyping
    Nightlight
    Notifications
    Phone
    PhoneCalls
    Pictures
    RemoteDesktop
    SharedExperiences
    SigninOptions
    Sound
    Tasks
    Touchpad
    Troubleshoot
    VideoPlayback
    Videos
    VoiceActivation
    WiFiCalling
    WindowsHelloFace
    WindowsHelloFingerprint
    WindowSecurity

YourPhone

New Commands:

LOCAL

Define variables that are local to a library function or to a batch file. The syntax is:

LOCAL *var1, var2, ...*

LOCAL will save the existing values of the specified environment variables (if any) and then delete the variable from the environment. You can then SET a new variable with that name; when the library function or batch file exits, the local variables are deleted from the environment and the previous values (if any) are restored.

SSHEXEC

The SSHEXEC command establishes a Secure Shell (SSH) connection to a server and starts up the user's default shell. Press Ctrl-C to disconnect from the other system. The syntax is:

SSHEXEC [/A /F *filename* /Gn /H *fwhost* /IPV6 /R *port* /S /T *type* /U *user* /P *password*] *host* /L *name*[:*pssword*] "*command* ..."

/A(utodetect firewall)
/F(ilename for host stdin)
/G (logging level)
/H (firewall host)
/IPv6
/L (user:password)
/P (firewall password)
/R(emote port)
/S(tatus messages)
/T (firewall type)
/U (Firewall user name)

host - Host name
command - Command to pass to the default host shell

If you don't specify a username, SSHEXEC 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, SSHEXEC will prompt for it.

If you want to do redirection on the remote system, enclose the command argument list in double quotes. The double quotes will be removed before passing the commands to the remote system.

SSHEXEC will display the host name & user name and prompt for a line of input, then send it to the host shell and return to the prompt to wait for the next line. SSHEXEC will display any output sent by the host to STDOUT and STDERR. When you type "exit" at the prompt, or the host disconnects SSHEXEC will exit.

VDESKTOP

Manage Windows 10 virtual desktops (requires Windows 10 build 21313 or later). VDESKTOP lets you create, remove, or switch desktops. The syntax is:

VDESKTOP [[/N="*name"]* /C [/W="file"] /R id /S [id] - +]

/C   Create a new desktop
/R   Remove the specified desktop. Id can be a desktop number (1 - n) or the GUID for that desktop.
/S   Switch to the specified desktop. If *id* isn't specified, switch to the desktop created with /C. *id* can either be a desktop number (1 - n) or the GUID for that desktop.

/N="*name*"   You can optionally specify a desktop name. If you don't specify a name, you need to use a desktop number (1 - *n*) or the desktop GUID. Note that with the current Windows builds, the name is not updated in the Task View, though it is usable with subsequent VDESKTOP commands, and it will be displayed properly when the system is restarted.
/W="*file*"   When used with /C, /W specifies the image file to use for the background wallpaper for the desktop. Note that with the current Windows builds, the background will not be updated until the system is restarted.

WMIRUN

Use WMI to run methods on a local or remote machine. You must be running in an elevated session. The syntax is:

WMIRUN /USER=*user* /PASSWORD=*password* /CLASS=*classname* /METHOD=*methodname networkresource command*

/USER=*username* - The user name to use for remote queries

/PASSWORD=*password* - The password to use for remote queries

/CLASS=*classname* - The WMI class name

/METHOD=*methodname* - The WMI method name

networkresource - WMI namespace. The namespace argument for remote servers will look something like "\\remote-server\root\cimv2" (substitute your server name for "remote-server").

command - The command you want WMI to run.

For example, this command terminates process 26568 on the local machine:

```
WMIRUN /method=Terminate /class=Win32_Process "\\.\root\CIMV2"
Win32_Process.Handle="26568"
```

2.5   What's New in Version 27

CMDebug 27.0:

The CMDebug installer is built with a new version of Advanced Installer.

The GUI framework library has been updated.

The Scintilla edit control has been updated to version 4.4.5..

Many performance & size improvements.

The debugger has a new "Call Stack" window that displays the current call stack (the filename, line #, command line, and the line(s) that called it (i.e., GOSUB or CALL). Double-clicking on a line in the Call Stack window will take you to that line in the tab window. (Note that the call stack is only expanded when you "Step Into" the next command.)

When debugging batch files that CALL or chain to another batch file (or GOSUB "filename: :label), or call a library function, if you "Step Into", the debugger will open a new tab window for the new file / library (if a tab window with that file doesn't already exist). The new tab window will be automatically closed when control returns to the calling batch file. If you stop debugging, all the windows will remain open to allow you to make edits.

The Environment window is rewritten after stepping through each command so it can catch updates. It now saves its row position and restores that after rewriting the contents (so it no longer always goes back to the beginning of the environment list).

TCC-RT:

The TCC-RT installer is built with a new version of Advanced Installer.

Dropped 32-bit support for TCC-RT v27.

Many performance & size improvements.

The IPWorks internet & compression libraries have been updated to IPWorks 2020.

The embedded Lua interpreter has been updated to version 5.4.2.

Added support for Python 3.9.

Batch file nesting limits have been removed (now only subject to the memory available).

Variable name length limits have been removed.

FTP - TCC will try to preserve timestamps when transferring files. The MDTM command is used when downloading, and the MFTM command is used when uploading. The server must support these commands for this to work.

Array variable expansion now supports arithmetic expressions (for example, "echo %myarray[%i*3]").

The previous XML parser in *TCC* (msxml6) has been replaced with a more powerful parser that provides much more capability, including both reading & writing XML and JSON files.

Help:

The help is built with a new version (8.2) of Help & Manual.

New TMCD.INI Directives:

PowerShellProfileID=*string* - The profile Id to look for when loading profiles. *TCC* will look for profiles that begin with the specified Id. For instance, the default value is "Microsoft.PowerShell" so the class will look for profiles named "Microsoft.PowerShell_profile.ps1".

TimeServerPort=*n* - The UDP port where the remote time server is listening (default 123). If TimeServerProtocol is set to tpTimeProtocol (1) the port will be set to 37.

TimeServerProtocol=*n* - The protocol used to connect to the time server. The default is 1 (tpSNTP). The Time protocol may be selected by setting this value to 0 (tpTimeProtocol).

New TCC-RT Internal Variables:

_BATCHPATH - Pathname of the current batch file (including the trailing \).

_IPDNSOTHER - A space-delimited list of other DNS servers configured for the host machine. (The primary server is returned by %_IPDNSSERVER.)

New TCC-RT Variable Functions:

@IPBROADCAST - The broadcast address of the specified network adapter.

@IPBROADCAST[*adapter*]

adapter - the index of the adapter

@IPDHCPENABLED - Returns 1 if the specified network adapter has DHCP enabled.

@IPDHCPENABLED[*adapter*]

adapter - the index of the adapter

@IPEXPIRES - The expiration date and time of the lease obtained by the specifed network adapter.

@IPEXPIRES[*adapter*]

adapter - the index of the adapter

@IPIPV6LL - The IPv6 link local address of the specified network adapter.

@IPIPV6LL[*adapter*]

adapter - the index of the adapter

@IPOBTAINED - The date and time of when the current lease was obtained by the network adapter.

@IPOBTAINED[*adapter*]

adapter - the index of the adapter

@IPOTHER - Returns a space-delimited list of alternate addresses for the specified host (if any). Most hosts have only one IP interface; this function is for querying multihomed hosts (hosts with more than one interface).

@IPOTHER[name, address]

name - the host name
address - the host address

@IPOTHERL - Returns a space-delimited list of any other IP addresses leased by the specified network adapter.

@IPOTHERL[*adapter*]

adapter - the index of the adapter

@IPSTATUS - Returns the current status of the specified network adapter.

@IPSTATUS[*adapter*]

adapter - the index of the adapter

Possible return values are:

Up
Down
Testing
Unknown
Dormant
NotPresent
LowerLayerDown

@IPWINSSERVER2 - Returns the secondary WINS server for the specified network adapter.

@IPWINSSERVER2[*adapter*]

adapter - the index of the adapter

@JSONCLOSE - closes a JSON file opened by @JSONOPEN. The syntax is:

@JSONCLOSE[]

@JSONENDARRAY - Writes the closing bracket of a JSON array. The syntax is:

@JSONENDARRAY[]

@JSONENDOBJECT - Writes the closing brace of a JSON object. The syntax is:

@JSONENDOBJECT[]

@JSONFLUSH - Flushes the JSON parser buffers. The syntax is:

@JSONFLUSH[]

@JSONHASXPATH - returns 1 if the XPath exists in the JSON file, 0 if it doesn't. The syntax is:

@JSONHASXPATH[*["filename",]path*]

@JSONINPUT - Parse an input string as JSON data. (This is used in place of @JSONOPEN.) The syntax is:

@JSONINPUT[*inputdata*]

@JSONINSERTPROPERTY - Writes a value of a property. The file must have been opened with a previous @JSONOPEN. The syntax is:

@JSONINSERTPROPERTY[xpath,name,*value,type,position*]

Name specifies the name of the property.

Value specifies the new value.

Type specifies the type of the value. Possible values are:
- 0 (Object)
- 1 (Array)
- 2 (String)
- 3 (Number)
- 4 (Bool)
- 5 (Null)
- 6 (Raw)

The *Position* parameter specifies the position of Value relative to the element specified by *XPath*. Possible values are:
- 0 (Before the current element)
- 1 (After the current element)
- 2 (The first child of the current element)
- 3 (The last child of the current element)

@JSONINSERTVALUE - Inserts the specified value at the selected position. The file must have been opened with a previous @JSONOPEN. The syntax is:

@JSONINSERTVALUE[xpath,*value,type,position*]

Value specifies the new value.

Type specifies the type of the value. Possible values are:
- 0 (Object)
- 1 (Array)
- 2 (String)
- 3 (Number)
- 4 (Bool)

- 5 (Null)
- 6 (Raw)

The *Position* parameter specifies the position of Value relative to the element specified by *XPath*. Possible values are:
- 0 (Before the current element)
- 1 (After the current element)
- 2 (The first child of the current element)
- 3 (The last child of the current element)

@JSONNODES - Returns the number of nodes (children) for the specified path in a JSON file. The syntax is:

   @JSONNODES[["*filename*",]*path*]

@JSONOPEN - open a JSON file for use by @JSONXPATH and/or @JSONNODES. The syntax is:

   @JSONOPEN[*filename*]

@JSONOUTPUT - Output JSON to a string after processing. (This is used in place of @JSONSAVE.) The syntax is:

   @JSONOUTPUT[]

@JSONPUTNAME - Writes the name of a property. The file must have been opened with a previous @JSONOPEN. The syntax is:

   @JSONPUTNAME[*name*]

@JSONPUTPROPERTY - Writes the name of a property and its value to a JSON file. The syntax is:

   @JSONPUTPROPERTY[*name,value,type*]

The *name* parameter specifies the name of the property.

The *value* parameter specifies the value of the property.

The *type* parameter specifies the type of the value. Possible values are:
- 0 (Object)
- 1 (Array)
- 2 (String)
- 3 (Number)
- 4 (Bool)
- 5 (Null)
- 6 (Raw)

@JSONPUTRAW - Writes a raw JSON fragment to a JSON file. The syntax is:

> @JSONPUTRAW[*text*]

@JSONPUTVALUE - Writes a value of a property. The file must have been opened with a previous @JSONOPEN. The syntax is:

> @JSONPUTVALUE[*value,type*]

Value specifies the new value.

ValueType specifies the type of the value. Possible values are:
- 0 (Object)
- 1 (Array)
- 2 (String)
- 3 (Number)
- 4 (Bool)
- 5 (Null)
- 6 (Raw)

@JSONREMOVE - Removes the element or value set in XPath. The syntax is:

> @JSONREMOVE[*xpath*]

If *xpath* is not specified, @JSONREMOVE will use the current XPath.

@JSONRESET - Flushes the JSON parser buffers, and initializes the parser to its default state. The syntax is:

> @JSONRESET[]

@JSONSAVE - Saves the modified JSON document to the specified output file. The file must have been opened with a previous @JSONOPEN. The syntax is:

> @JSONSAVE[*outputfile*]

@JSONSETNAME - Sets a new name for the element specified by XPath. The file must have been opened with a previous @JSONOPEN. The syntax is:

> @JSONSETNAME[*xpath,] name*]

If *xpath* is not specified, @JSONSETNAME will default to the current *xpath*.

@JSONSETVALUE - Sets a new value for the element specified by XPath. The file must have been opened with a previous @JSONOPEN. The syntax is:

> @JSONSETVALUE[*xpath,]value,type*]

If *xpath* is not specified, @JSONSETVALUE will default to the current *xpath*.

value specifies the new value.

type specifies the type of the value. Possible values are:

- 0 (Object)
- 1 (Array)
- 2 (String)
- 3 (Number)
- 4 (Bool)
- 5 (Null)
- 6 (Raw)

@JSONSTARTARRAY - Writes the opening bracket of a JSON array. The syntax is:

@JSONSTARTARRAY[]

@JSONSTARTOBJECT - Writes the opening brace of a JSON object. The syntax is:

@JSONSTARTOBJECT[]

@JSONXPATH - JSON XPath query. The syntax is:

@JSONXPATH[*["filename",]path*]

The *path* is a series of one or more element accessors separated by '/'. The path can be absolute (starting with '/') or relative to the current XPath location. Note: When using XPath notation the root element is always referred to as "json". This means all paths will begin with "/json".

The following are possible values for an element accessor:

name	A particular element name.
[i]	The i-th subelement of the current element.
..	the parent of the current element.

@PUNYDECODE - Decode a Punycode string or file. The format is:

@PUNYDECODE[**s**,*string*]
@PUNYDECODE[*inputfile*,*outputfile*]

@PUNYENCODE - Encode a Punycode string or file. The format is:

@PUNYENCODE[**s**,*string*]
@PUNYENCODE[*inputfile*,*outputfile*]

@QPDECODE - Decode using the Quote-Printable MIME format (using only special characters). The format is:

@QPDECODE[**s**,*string*]
@QPDECODE[*inputfile*,*outputfile*]

@QPENCODE - Encode using the Quote-Printable MIME format (using only special characters). The format is:

@QPENCODE[**s**,*string*]
@QPENCODE[*inputfile*,*outputfile*]

@XMLENDELEMENT - Writes the closing tab of an XML element opened using @XMLSTARTELEMENT. If no elements are open, @XMLENDELEMENT returns an error. The syntax is:

@XMLENDELEMENT[]

@XMLFLUSH - Flushes the XML parser buffers, and checks its end state. The syntax is:

@XMLFLUSH[]

@XMLGETATTR - Returns the value of the specified attribute. The syntax is:

@XMLGETATTR[*["filename",]*,*attributename*]

@XMLHASXPATH - returns 1 if the XPath exists in the XML file, 0 if it doesn't. The syntax is:

@XMLHASXPATH[*["filename",]path*]

@XMLINPUT - Parse an input string as XML data. (This is used in place of @JXMLOPEN.) The syntax is:

@XMLINPUT[*inputdata*]

@XMLOUTPUT - Output XML to a string after processing. (This is used in place of @XMLSAVE.) The syntax is:

@XMLOUTPUT[]

@XMLPUTCDATA - Writes an XML CDATA block. The file must have been opened with a previous @XMLOPEN. The syntax is:

@XMLPUTCDATA[*text*]

@XMLPUTCOMMENT - Writes an XML comment block. The file must have been opened with a previous @XMLOPEN. The syntax is:

@XMLPUTCOMMENT[*text*]

@XMLPUTELEMENT - Writes a simple XML element with no attributes and the specified value between the opening and closing tags. The syntax is:

@XMLPUTELEMENT*name,namespaceURI, value*]

If *name* is a local name without a prefix, *TCC* will automatically introduce a new xmlns="NamespaceURI" attribute if necessary.

If *name* is in the form prefix:local, then *TCC* will automatically introduce a new xmlns:prefix="NamespaceURI" as necessary.

When calling @XMLPutElement or @XMLStartElement, if a *namespaceURI* is not specified an empty namespace will be defined for the element. If a namespace should be associated with the element, a *namespaceURI* value must be provided. When creating the XML, *TCC* will determine if the namespace already exists to avoid duplicate definitions of the same namespace.

@XMLPUTSTRING - Writes text inside an XML element. The file must have been opened with a previous @XMLOPEN. The syntax is:

@XMLPUTSTRING[*text*]

@XMLREMOVECHILDREN - Removes the children of the element at the specified (or current) XPath. The element itself remains. The syntax is:

@XMLREMOVECHILDREN[*[path]*]

@XMLREMOVEELEMENT - Removes the element and its children at the specified (or current) XPath. The syntax is:

@XMLREMOVEELEMENT[*[path]*]

@XMLRESET - Flushes the XML parser buffers, and initializes the parser to its default state. The syntax is:

@XMLRESET[]

@XMLSAVE - Saves the modified XML document to a the specified output file. The file must have been opened with a previous @XMLOPEN. The syntax is:

@XMLSAVE[*outputfile*]

@XMLSTARTELEMENT - Writes the opening tag of a new XML element. If an XML element is already opened, then this element is written as a child. The syntax is:

@XMLSTARTELEMENT*name,namespaceURI*]

Updated TCC-RT Internal Variables:

_cpu is obsolete and has been removed.

_wow64 is obsolete and has been removed.

Updated TCC-RT Variable Functions:

@PING - added two new options for the time to live and the ICMP service type.

@ping[host[,timeout[,size[,ttl[,type]]]]]

ttl - Time to live - defaults to the TTL value of the underlying TCP/IP subsystem

type - ICMP service type (default 8)

@SELECT - if you set the *sort* option to -1, @SELECT will sort the list in reverse order.

@XMLXPATH - the *path* argument has additional options.The path is a series of one or more element accessors separated by '/'. The path can be absolute (starting with '/') or relative to the current XPath location.

The following are possible values for an element accessor:

'name'	A particular element name
name[i]	The i-th subelement of the current element with the given name
[i]	The i-th subelement of the current element
[last()]	The last subelement of the current element
[last()-i]	The subelement located at the last location minus i in the current element
name[@attrname="attrvalue"]	The subelement containing a particular value for a given attribute (supports single AND double quotes)
..	The parent of the current element

Updated TCC-RT Commands:

COPY

COPY /G now supports HTTP / HTTPS copies.

/GZ - When copying to an HTTP / HTTPS target, this option will compress the file into gzip format before uploading it.

ESET

/K*"::regex"* - defines a regular expression mask for the input.

If you are editing a typed environment variable (see SET /T), ESET will create a matching regular expression mask for the input.

HASH

HASH now supports ranges and selection by attributes.

/S - Return hashes for matching files in the current directory and subdirectories.

IFTP

IFTP will try to preserve timestamps when transferring files. The MDTM command is used when downloading, and the MFTM command is used when uploading. The FTP host server must support these commands for this to work.

LIBRARY

(Not new, but apparently never documented.) The command line following the library function name is passed to the function, and the arguments can be referenced with the same %1 - %n syntax as used by batch files and aliases.

LUA

The internal Lua support has been updated to Lua 5.4.2.

PAUSE

/T - Displays a countdown timer. Must be used with /W*n*, which must precede /T.

SET

Array variable assignment and display now support arithmetic expressions (for example, "set %myarray[%i*3]=somevalue").

SETARRAY

You can initialize arrays by appending [*value*] to the definition. For example, to initialize all of the array elements to 0:

    setarray myarray[100] [0]

TPIPE

The TextPipe engine has been updated to version 11.7.5.

New TCC-RT Commands:

COMMANDS

Display, enable, or disable the *TCC* internal commands. The syntax is:

    COMMANDS [/D /E /P] *commandname ...*

If you do not enter any arguments, COMMANDS will display all of the internal commands. Disabled commands will be enclosed in parentheses. If you enter command names without a /D or /E, COMMANDS will show the current state of those commands.

   /D - Disable one or more commands. If you do not provide any command names, COMMANDS will display all of the disabled commands.

   /E - Enable one or more commands. If you do not provide any command names, COMMANDS will display all of the enabled commands.

   /P - Pause after displaying each page.

DNS

Display the DNS records for the specified DNS server and host domain. The syntax is:

    DNS [/Nh] *server hostname*

      /Nh - Don't display the columns header

      server - The address of the DNS server

---

hostname - The host domain to query

For example:

DNS 1.1.1.1 jpsoft.com

THREAD

Execute a command in a separate thread. The syntax is:

THREAD *command [args]*

It is the user's responsibility to ensure that there are no I/O or file system conflicts when running multiple THREAD commands and/or running THREAD simultaneously with commands in the primary TCC thread.

THREAD will set the internal variable _thread_result to the return value of *command*.

TOAST

Displays Windows Toast notifications, a popup window that appears on the lower right corner of the display. Unlike message boxes, Toast popups are not modal and will disappear after a few seconds. Windows will not display Toast notifications if the user has disabled notifications, either for *TCC* or everywhere.

The syntax is:

TOAST /template=*n* /text1="*text*" [*options*]

TOAST sets two internal command variables:

_toast
    0 - no toast active or no user response yet
    1 - user clicked on the toast
    2 - user dismissed the toast
    3 - toast timed out
    4 - application hid the toast
    5 - toast was not activated
    6 - toast failed
    7 - system does not support toasts
    8 - unhandled option
    9 - multiple texts were provided
    10 - toast notification manager initialization failure
    11 - toast could not be launched

_toast_action
    0 - user has not clicked on a button
    1 - user clicked on first button
    2 - user clicked on second button
    3 - user clicked on third button

The TOAST command exits after calling Windows to display the Toast notification. Windows will call back to TOAST with the Toast results and actions, so the _toast and _toast_action variables will not be set until the user either clicks on the Toast or it times out.

The TOAST options are:

/action="*text*" - You can have one or more actions. Each action creates a button on the Toast window; clicking on that button will set the _toast_action internal variable.

/attribute="*text*" - Attribution text displayed on the bottom of the Toast window.

/audio=*n* - Windows system sound to play when the notification is displayed.
    0 - DefaultSound
    1 - IM
    2 - Mail
    3 - Reminder
    4 - SMS
    5 - Alarm
    6 - Alarm2
    7 - Alarm3
    8 - Alarm4
    9 - Alarm5
    10 - Alarm6
    11 - Alarm7
    12 - Alarm8
    13 - Alarm9
    14 - Alarm10
    15 - Call
    16 - Call1
    17 - Call2
    18 - Call3
    19 - Call4
    20 - Call5
    21 - Call6
    22 - Call7
    23 - Call8
    24 - Call9
    25 - Call10

/audiostate=*n* - Specifies whether you want to display the sound (see /audio above) once, looping, or not at all.
    0 - Default
    1 - Silent
    2 - Loop

/duration=*n* - The time to display the Toast notification
    0 - Default
    1 - Short
    2 - Long

/expire=*n* - Number of seconds before the notification expires.

/image="*pathname*" - The image file you want to display (for template types 0 - 3)

/template=*n* - The type of Windows Toast you want to display:
    0 - An image on the left, and a string that occupies a maximum of three lines
    1 - An image on the left, a bold string on the first line and a second string wrapped across the second and third lines.
    2 - An image on the left, a bold string on the first and second lines, and a second string on the third line.

3 - An image on the left, a bold string on the first line, a second string on the second line, and a third string on the third line.
4 - A string that occupies a maximum of three lines
5 - A bold string on the first line and a second string wrapped across the second and third lines.
6 - A bold string on the first and second lines, and a second string on the third line.
7 - A bold string on the first line, a second string on the second line, and a third string on the third line.

/S - Create the shortcut to *TCC* required for Toast notifications. (Not valid with any other options.) This is normally done by the installer, so you shouldn't need to run TOAST /S unless the shortcut was removed.

/text1="*text*" - Text to display in the first line (template types 0 - 7).

/text2="*text*" - Text to display in the second line (for template types 1, 2, 3, 5, 6, and 7)

/text3="*text*" - Text to display in the third line (for template types 3, and 7)

2.6   What's New in Version 26

<u>**CMDebug 26.0:**</u>

Windows 7 support has been dropped from all products (Take Command, TCC, CMDebug, and TCC-RT).

Except for TCC-RT, 32-bit Windows support is deprecated in v26. The v26 CMDebug installer is 64-bit only. 32-bit installers will be available on request for multisystem licenses.

The CMDebug and TCC-RT installers are built with a new version of Advanced Installer.

The GUI framework library has been updated.

The IDE will now scale properly when moving between monitors with different DPI values.

The Scintilla edit control has been updated to version 4.3.2.

Many performance & size improvements.

Expanded the help for the IDE / Debugger.

The IDE will monitor the filesystem for any changes to the file(s) being edited. If another application modifies a file, the IDE will display a message notifying you of the change and asking if you want to reload the updated file.

Changed all of the IDE icons to a modern "Fluent" design.

Most of the IDE menu entries have icons, giving you more options to customize the toolbar.

Improved the font display slightly.

There is a new startup option following the file name to goto a line number:

/gotoline:*nn*

For example:

bdebugger mytest.cmd /gotoline:24 *[batch file arguments...]*

There is a new option "Syntax Colors" in the Options menu that allows you to select the colors used in the syntax colorization from a 16 million color palette. When you click on one of the foreground or background color buttons, the IDE will display a color picker dialog to let you choose colors.

The Error Lookup dialog (Debug / Error Lookup) now supports NTSTATUS error codes.

The IDE has additional tabs for Local Aliases, Global Aliases, Local Functions, and Global Functions. They will only be displayed if the appropriate local / global list exists.

There is a new option in the File menu to reload the file in the current tab from the disk.

There is a new option in the File menu to delete the file in the current tab to the recycle bin.

There are two new options in the File menu to save & load sessions. You create a session with the "Save Session" menu option, which creates a file with the names of the files in the tab windows. "Load Session" will open the files in the *.session file you specify.

There is a new option in the Options menu to set the caret color for the tab edit windows.

There is a new option in the Options menu to change the working directory for the IDE.

There is a new submenu in the Options menu to select the window tab location (top, bottom, left, or right).

The context menu on the tab labels has a new entry "Copy Full Path" that copies the full pathname of the file in that tab to the clipboard.

The context menu on the tab labels has a new entry "Close All" that closes all of the tab windows.

The "Tabs..." menu entry has been moved from Options to Edit.

The IDE dialogs have been tweaked to be cleaner and more readable.

The tooltip for the transparency slider (lower right corner) now displays the current transparency setting (20 - 255; higher values are more opaque).

The Batch Arguments combo box on the IDE toolbar now displays hint text if you don't supply arguments at startup.

Dragging text in the a edit window will now automatically scroll the window when you reach the edges.

The Unicode value on the statusbar for the character at the cursor location now supports UTF8 multibyte characters.

TCC-RT:

The language dll's are substantially smaller and load faster.

Many performance & size improvements.

The embedded Lua interpreter has been updated to version 5.4.

Python support has been rebuilt with the new 3.8.2 release.

Added support for Windows Server 2019.

Most of the remaining string size limits have been removed (except for those where the Windows APIs have limits).

Help:

The help is built with a new version of Help & Manual 8.

New TCC-RT Internal Variables:

_osbuildex - Returns the Windows build number + the sub-build number (for example, "19041.84").

New TCC-RT Variable Functions:

@DATEFMT - Formats a date/time in a custom format. The syntax is:

@DATEFMT[*date,format*]

date - The date to format (in yyyy-mm-dd hh:mm:ss format). If *date* is *, @DATEFMT defaults to the current date/time. Valid dates are January 1, 1970 (1970-1-1) to December 31, 3000 (3000-12-31). The time must be in 24-hour format.

format - The custom format to use. (Note that the %'s will normally need to be doubled or escaped to prevent TCC from expanding them before @DATEFMT sees them.) The formatting options are:

Code	Replacement string
%a	Abbreviated weekday name in the locale
%A	Full weekday name in the locale
%b	Abbreviated month name in the locale
%B	Full month name in the locale
%c	Date and time representation in the locale
%C	The year divided by 100 and truncated to an integer, as a decimal number (00–99)
%d	Day of month as a decimal number (01 - 31)
%D	Equivalent to **%m/%d/%y**
%e	Day of month as a decimal number (1 - 31), where single digits are preceded by a space
%F	Equivalent to **%Y-%m-%d**
%g	The last 2 digits of the ISO 8601 week-based year (00 - 99)

%G	The ISO 8601 week-based year as a decimal number
%h	Abbreviated month name (equivalent to **%b**)
%H	Hour in 24-hour format (00 - 23)
%I	Hour in 12-hour format (01 - 12)
%j	Day of the year as a decimal number (001 - 366)
%m	Month as a decimal number (01 - 12)
%M	Minute as a decimal number (00 - 59)
%n	A newline character (**\n**)
%p	The locale's A.M./P.M. indicator for 12-hour clock
%r	The locale's 12-hour clock time
%R	Equivalent to **%H:%M**
%S	Second as a decimal number (00 - 59)
%t	A horizontal tab character (**\t**)
%T	Equivalent to **%H:%M:%S**, the ISO 8601 time format
%u	ISO 8601 weekday as a decimal number (1 - 7; Monday is 1)
%U	Week number of the year as a decimal number (00 - 53), where the first Sunday is the first day of week 1
%V	ISO 8601 week number as a decimal number (00 - 53)
%w	Weekday as a decimal number (0 - 6; Sunday is 0)
%W	Week number of the year as a decimal number (00 - 53), where the first Monday is the first day of week 1
%x	Date representation for the locale
%X	Time representation for the locale
%y	Year without century, as decimal number (00 - 99)
%Y	Year with century, as decimal number
%z	The offset from UTC in ISO 8601 format; no characters if time zone is unknown
%Z	Either the locale's time-zone name or time zone abbreviation, depending on registry settings; no characters if time zone is unknown
%%	Percent sign

Characters that do not begin with a % are displayed unchanged.

The **#** flag may prefix any formatting code. In that case, the meaning of the format code is changed as follows:

Format code	Meaning
%#a, **%#A**, **%#b**, **%#B**, **%#g**, **%#G**, **%#h**, **%#n**, **%#p**, **%#t**, **%#u**, **%#w**, **%#X**, **%#z**, **%#Z**, **%#%**	**#** flag is ignored.
%#c	Long date and time representation, appropriate for the locale. For example: "Wednesday, March 25, 2020, 12:41:29".
%#x	Long date representation, appropriate to the locale. For example: "Wednesday,

March 25, 2020".

%#d, **%#D**, **%#e**, **%#F**, **%#H**, **%#I**, | Remove leading zeros or spaces (if any).
%#j, **%#m**, **%#M**, **%#r**, **%#R**, **%#S**,
%#T, **%#U**, **%#V**, **%#W**, **%#y**, **%#Y**

The ISO 8601 week and week-based year produced by **%V**, **%g**, and **%G**, uses a week that begins on Monday, where week 1 is the week that contains January 4th, which is the first week that includes at least four days of the year. If the first Monday of the year is the 2nd, 3rd, or 4th, the preceding days are part of the last week of the preceding year. For those days, **%V** is replaced by 53, and both **%g** and **%G** are replaced by the digits of the preceding year.

@FILETYPE - returns the encoding type of the file. The syntax is:

@FILETYPE[*filename*]

You must enable UTF8 input for TCC to recognize UTF8 files; see OPTION / Setup. The possible return values are:

ASCII
UTF8
UTF16

Updated TCC-RT Internal Variables:

 _DOS - Added support for Windows Server 2019.

Updated TCC-RT Variable Functions:

@CRC32 - if the first argument is "sa", treat the second argument as an ASCII string. If the first argument is "s8", treat the argument as a UTF8 string.

@EVAL - has an optional + argument at the end of the precision string (i.e., %@eval[nnn=x.y+] to specify that positive reaults should be prefixed by a +.

@HISTORY - has an optional third argument specifying whether you want the local history list or the global history list. (If you want to specify the history list to use, but not the (optional second argument) word to return, set *word* to -1.)

@HISTORY[*entry[*, *word*, *[L | G]]*]

@MD5- if the first argument is "sa", treat the second argument as an ASCII string. If the first argument is "s8", treat the argument as a UTF8 string.

@SHA1- if the first argument is "sa", treat the second argument as an ASCII string. If the first argument is "s8", treat the argument as a UTF8 string.

@SHA256- if the first argument is "sa", treat the second argument as an ASCII string. If the first argument is "s8", treat the argument as a UTF8 string.

@SHA384- if the first argument is "sa", treat the second argument as an ASCII string. If the first argument is "s8", treat the argument as a UTF8 string.

@SHA512- if the first argument is "sa", treat the second argument as an ASCII string. If the first argument is "s8", treat the argument as a UTF8 string.

Updated TCC-RT Commands:

ALIAS

ALIAS now supports both local and global lists simultaneously. If you have both local and global lists defined, when displaying, creating or deleting aliases, you can specify which list you want ALIAS to use.

/GL - use the global alias list

/LL - use the local alias list

If you have both local and global lists defined and do not specify /GL, ALIAS will default to using the local list.

TCC will first look for aliases in the local list; if not found TCC will search the global list.

If you use the /G option to convert a local alias list to a global alias list, ALIAS will not do the conversion if a global alias list already exists (for example, in another TCC session or in SHRALIAS).

COLOR

Added new options for setting the foreground and background color in a TCC console window (not a TCMD tab window) to 16 million or 256 colors. You must be running Windows 10 and have ANSI enabled.

/FG *r,g,b* - sets the foreground color to the 16 million color RGB value specified. Valid ranges for r, g, and b are 0-255.

/BG *r,g,b* - sets the background color to the 16 million color RGB value specified. Valid ranges for r, g, and b are 0-255.

/FG *color* - sets the foreground color to the 256 color (xterm) value specified. Valid range for *color* is 0 - 255.

/BG *color* - sets the background color to the 256 color (xterm) value specified. Valid range for *color* is 0 - 255.

/P [*color]* - displays a color picker dialog to select a color. Must be used with /FG or /BG, and cannot be combined with /F.

COPY

COPY /S will now display the number of directories copied (if any).

/CDA - copy the attributes from each of the source subdirectories to the target subdirectories.

DATE

DATE has a new option *"datefmt"* that displays the current date/time in a custom format. The formatting characters are the same as used by the @DATEFMT function (see above). The DATE syntax is:

DATE [/Fn /T /U *"format"*] [mm-dd-yy]] [AM | PM]

DEL

DEL /S /X will now display the number of directories removed (if any).

DELAY

DELAY hh:mm:ss or DELAY mm:ss will wait for the specified amount of time.

DIR

/CD:"*colordir*" - define a customized directory colorization string to use instead of the COLORDIR environment variable or the ColorDir option in TCMD.INI.

DO

DO will delete any temporary files created by using CLIP: in a DO statement.

ENDLOCAL

If you only have global aliases defined, ENDLOCAL will restore the global list saved by SETLOCAL (as in previous versions). If you have both local aliases and global aliases defined, ENDLOCAL will only restore the local list that was saved by SETLOCAL. See also SETLOCAL.

ESET

/LL (or /LL) - use the global alias or function list. ESET will default to using the local list if it exists; if it doesn't ESET will look for a global list.

/G (or /GL) - use the local alias or function list.

EXCEPT

/NM - if no match is found for the argument(s) in the exception list, EXCEPT will not execute the command.

FUNCTION

FUNCTION now supports both local and global lists simultaneously. If you have both local and global lists defined, when displaying, creating or deleting functions, you can specify which list you want FUNCTION to use.

/GL - use the global function list

/LL - use the local function list

If you have both local and global lists defined and do not specify /GL, FUNCTION will default to using the local list.

TCC will first look for a user-defined function in the local list; if not found TCC will search the global list.

If you use the /G option to convert a local function list to a global function list, FUNCTION will not do the conversion if a global function list already exists (for example, in another TCC session or in SHRALIAS).

LIST

LIST now supports paging backwards through piped input.

MEMORY

MEMORY now shows the local and global alias and function sizes.

MOVE

MOVE /S will now display the number of directories moved (if any).

/MDA - copy the attributes from each of the source subdirectories to the target subdirectories. (Only valid if moving to another drive; otherwise MOVE does a rename of the top-level directory and all of the subdirectory attributes are retained.)

MSGBOX

If TCC is running in a Take Command tab window, the message box will be centered on the tab window.

/PC - center the message box on the desktop.

/X - the message box cannot be moved.

PDIR

/CD:"*colordir*" - define a customized directory colorization string to use instead of the COLORDIR environment variable or the ColorDir option in TCMD.INI.

QUERYBOX

Removed the maximum length limit (previously 255 characters) for the input string.

Removed the maximum length limit (previously 127 characters) for the /CUE string.

REGDIR

/Nb - Don't display the contents of REG_BINARY values.

/TS - include seconds in the last write time display.

REGDIR will now display all the strings in a REG_MULTI_SZ.

REN

REN /S will now display the number of directories renamed.

SETLOCAL

If you only have global aliases defined, SETLOCAL will behave as in previous versions and temporarily copy the global list to a local list, and restore the global list on an ENDLOCAL. If you have both local aliases and global aliases defined, SETLOCAL will only save the local list, which will be restored by ENDLOCAL.

SHORTCUT

SHORTCUT will not try to fully qualify the command, startup directory, or link file name if they contain %'s. This allows you to embed variables in those arguments that will be expanded by Windows.

TEE

/F*"datefmt"* - prefix each line with a timestamp using a custom format. The formatting characters are the same as used by the @DATEFMT function (see above).

TIME

TIME has a new option *"datefmt"* that displays the current date/time in a custom format. The formatting characters are the same as used by the @DATEFMT function (see above). The TIME syntax is:

TIME [/S [server] /T /U *"format"*] [hh[:mm:ss]]] [AM | PM]

TPIPE

Updated the TextPipe Engine version from 9.9.4 to 11.6.

The TextPipeEngine dll is now 64-bit (for the x64 version of TCC). (The initial load is a bit slower, but everything runs faster.)

Updated Unicode compose/decompose functions for NFC, NFD, NFKC, NFKD.

Updated PDF libraries.

Updated code page converted library.

Upgraded regular expression library.

Unicode support upgraded to Unicode 12.1.

New option for Add Line Numbers (to reset at the start of a new file).

The Line Number filter has a new option:

/line=StartNumber,Increment,SkipBlank,DontNumberBlank,NumberFormat[,DontReset[,ResetNewFile]]

ResetNewFile - if 1, reset the count at the start of a new file. The default is 0.

Added a new selection filter:

/selection2=type, columnSpec, moveTo, processIndividually, excludeDelimiter, excludeQuotes, delimiter, customDelimiter, hasHeader

Type - the type of filter to add

0 Delete column
1 Restrict lines
2 Restrict columns
3 Restrict to bytes
4 Restrict to delimited fields (CSV, Tab, Pipe etc)
5 - unused
6 Remove lines
7 Remove delimited fields (CSV, Tab, Pipe etc)
9 Move columns
10 Move delimited fields (CSV, Tab, Pipe etc)
12 Copy columns
13 Copy delimited fields (CSV, Tab, Pipe etc)
17 Remove Byte Range
18 Extract fields

columnSpec - the double-quoted list of items to remove e.g. "1..10, 16, 20"

moveTo : integer - where to move or copy the columns or fields to. Default 1.

processIndividually - whether or not to apply sub filters to each CSV or Tab field individually, or to the fields as one string value. Default false.

excludeDelimiter - whether or not to include the comma or Tab field delimiter when passing the field to the sub filter. Default true.

excludeQuotes - whether or not to include the CSV quotes that may surround the field when passing the field to the sub filter. Default true.

delimiter - (optional) the index of the standard delimiter to use, or 6 for custom, default 0 for CSV

customDelimiter - (optional) the double quoted custom delimiter to use, default blank

hasHeader - (optional) true if the file's first row is a header row, default false.

The End of Line filter has two new options:

/eol=Input,Output,Length,**LFString,Remove**

LFString - the new line feed string on output when option 4 is chosen for Input
Remove - whether to remove bad EOLs (default 1)

Column specifications for Delimited field delete, extract and restrict can now specify multiple columns in one filter e.g. 6, 9, 61..63.

Added JSON output format to database filter.

/database=Mode,...

Mode = 4 - JSON

Added Convert Tab to JSON and Convert JSON to Tab filters.

/simple=type

type = 86 - Convert JSON to Tab
type = 87 - Convert Tab to JSON

Added Convert Word documents to RTF filter.

/Simple=type

type = 88 - Convert Word documents to RTF

New Sort by UTF-8 (case sensitive and insensitive).

/sort=Type,...

Type = 9 - UTF8 sort (case insensitive)
Type = 10 - UTF8 sort (case sensitive)

/InputClipboardUnicode=[0|1] - In clipboard mode, controls whether the input is handled as ANSI or Unicode. The default is 0 (ANSI).

Added options to the /Split= filter:

/split=type,SplitSize,SplitChar,SplitCharPos,SplitCharCount,SplitLines,SplitFilename[,FirstFileNumber[,PreventOverload]]

FirstFileNumber - (optional) the number of the first file, default 0

PreventOverload - (optional) true to prevent more than 10,000 files in one folder, default false

Defaults to UTF8 encoding instead of ANSI when loading/saving files.

Enhanced perl regex filter to allow Unicode characters to delimit Whole Words.

Updated case changing filters to work with UTF-8 encoded text.

Changed Convert Word/Excel/PDF to text filters to output UTF-8 text.

Remove BOM filter now detects if it has changed the file or not, and handles UTF-16 LE properly.

Search/replace list filters now support Unicode.

Improved Line Numbering filter to correctly tell the difference between start of file and start of restriction.

Improved error handling for specification filters that do not support multiple ranges.

Improved error handling for script filters and loading settings and Languages.

Field specification filters now support field names with embedded hyphens (-), and field names with spaces can be used by surrounding them with quotes.

A warning is now output when a field specification does not match the field names found in a file.

Improved the speed of "exact match" search/replace.

Text to Word List now recognises English possessives (or other abbreviations) ending with 's.

Updated Remove Blanks from Start of Line/End of Line to handle UTF-8 e3 80 80 IDEOGRAPHIC SPACE (common in Chinese text).

Split file filters will now remove the last file if it has zero bytes.

Split filter now processes macros after the file numbering has taken place.

Database filters now change the output extension to match the format.

The "Extract URL" filter now copes with any scheme, from the previously supported mailto:, http:, https:, nntp:, gopher:, ftp:, ftps:, and newer ones such as call: and skype:

Enhanced log output to provide information for every filter type (very useful for filter debugging).

TPIPE now checks for 'zombie' filters that follow a T-filters' secondary output filter (these filters do nothing).

New filter to convert Word documents to RTF.

HTML entities are now case sensitive.

The log now includes filter icons for easier identification.

Conversion from CSV to Tab now eliminates unnecessary quotes.

OpenOffice support for ODT, ODS, and ODP.

TREE

Added support for colorizing the output of TREE. The colorizing options and format are the same as DIR, and can include:

Extension
File attribute
File size
File date/time
Executable type
Ranges

VER

/C - displays the version information in the same format as CMD (i.e., "Microsoft Windows [Version 10.0.19559.1000]").

Y

There are two new options for timestamping the STDIN lines that Y writes to STDOUT:

/D - Prefix each line with the current date (in yyyy-mm-dd format).

/F"*format*" - a custom time/date *format* string. See @DATEFMT (above) for details on the *format* arguments.

/T - Prefix each line with the current time (in hh:mm:ss.ms format).

WMIQUERY

/Q - prevents the display of the property name when displaying properties.

New Commands:

CHRONIC

CHRONIC runs a command and hides its STDOUT and STDERR output unless the command fails. If the command succeeds, no output is displayed. The syntax is:

CHRONIC [/R] *command ...*

/R - Display the output if the command writes to STDERR. If /R is not specified, CHRONIC will only display the output if the command returns a non-zero exit code.

CHRONIC will display the STDOUT and STDERR output separately. For example:

c:\> CHRONIC *testcommand*
Exit code: 2
STDOUT:
stdout output here ...
STDERR:
stderr output here ...

PEE

PEE is similar to TEE, but instead of redirecting STDOUT to multiple files, it redirects it to multiple secondary commands via pipes. The syntax is:

PEE /D /F"*format*" /R /T *app ...*

/D - prefix each line with the current date

/F"*format*" - a custom time/date *format* string. See @DATEFMT (above) for details on the *format* arguments.

/R - redirect STDERR too

/T - prefix each line with the current time

SPONGE

SPONGE reads standard input and writes it to the specified file. Unlike output redirection, SPONGE reads all its input before opening the output file. This allows constructing pipes that read from and write to the same file. SPONGE reads standard input into a memory buffer, so piping extremely large amounts of data (i.e., multiple gigabyte) is not recommended.

The syntax is:

SPONGE [/A] *outputfilename*

/A - append output to *outputfilename*. The default is to overwrite *outputfilename*.

TS

TS reads lines from STDIN, prefixes a date/time stamp, and writes the line to STDOUT. TS is intended to be used in pipes, when you need to know when each line was received. The syntax is:

TS [/D /T "*format*"]

/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).

"..." - The optional *format* string. See @DATEFMT (above) for details on the *format* arguments.

If you don't specify any options, TS defaults to /D /T.

2.7   What's New in Version 25

CMDebug 25.0:

The Scintilla edit control has been updated to version 4.2.0.

Redrawing in the edit windows is smoother and faster.

Improved the load and save times for large files.

When loading a file, CMDebug will first check for the file type (UTF-16, UTF-8 with BOM, or ANSI). If the file doesn't have a UTF-16 or UTF-8 BOM, it is read as an ANSI file with the current console code page, and converted to UTF-8 before editing. It will be converted back to an ANSI file with the current code page when it is saved. This allows CMDebug to properly display high-bit ASCII characters in the editor.

The batch debugger has a new "Command Expansion" window that will pop up above the tab window when you start debugging. The Command Expansion window will show the original command line, the command line after alias expansion, and the command line after variable expansion. The Command Expansion window is a docking window, so it can be moved & attached at other locations. If you don't want to see the Command Expansion window, you can turn it off from the CMDebug "View / Command Expansion" menu option.

You can now single-step into command groups and FOR loops. Click on the "Step Into" button on the CMDebug toolbar. You will see the current command line being executed in the "Command Expansion" window (see above).

The "Modified" tab has a new column "Previous" that shows the previous value of the variable that was just changed.

When debugging, the CMDebug window will now keep the current line centered on the screen (unless it's on the last page). This allows you to see both the last few lines and the next ones to be executed.

You can change the CMDebug window transparency with Ctrl-Shift-Mousewheel.

The Watch, Modified, and Breakpoint windows will now save the column widths if you change them, and use the new widths when you restart CMDebug.

The edit window will now keep the current line highlighted even when not in focus.

The edit window will default to maintaining the same indentation as the previous line. The default can be changed with the MaintainIndent option in TCMD.INI.

Regular Expression searching (Find dialog) now uses the C++11 regular expression library instead of the previous limited regular expression support.

The profiler timer now uses the Windows performance counters. The resolution is now in milliseconds (.001 seconds) instead of hundredths (.01 seconds).

If you're using TCC syntax (not CMD), and the first command on the line is an internal TCC command, CMDebug will display the quick usage help on the status bar.

Added a new submenu to the File menu:

Encoding	Files are always treated as UTF-8 inside the editor. This option allows you to specify how the file will be written when it is saved to disk.
Default Codepage	When the file is saved it will be written using the current codepage
UTF16 Little Endian	When the file is saved it will be written as UTF-16
UTF8	When the file is saved it will be written as UTF-8
UTF8 with BOM	When the file is saved it will be written as UTF-8 with a leading BOM

Added a new entry to the Edit menu:

Copy+Append	Append the current selection to the existing clipboard contents.

Added a new submenu to the Edit menu:

End of Line Characters

CR + LF	Lines end in a Carriage Return + Linefeed (Windows default)
CR	Lines end in a Carriage Return (OSX default)
LF	Lines end in a Linefeed (Linux default)

Added a new entry to the Edit / Advanced menu:

Toggle current fold - toggles folding the current line on & off

Added two new folding entries to the View menu. (These will be a bit confusing if you don't turn on the folding margin in the Options menu!)

Toggle current fold - toggles folding the current line on & off
Toggle all folds - toggles every fold in the file

Added a new entry to the Debug menu:

Evaluate Command - runs the specified command in the context of the currently executing batch file. The output is displayed in a scrollable read-only edit control. Note that the command you run may change the result of the batch file being debugged.

The tab windows right click context menu has a new option: "Copy+Append" will append the current selection to the existing clipboard contents.

A Ctrl-Shift-C key will append the current selection to the existing clipboard contents.

The Regular Expression Analyzer (Tools / Regular Expressions...) now has a microsecond timer (to the right of the "Test" edit control) that measures the time it took to evaluate the expression.

The Regular Expression Analyzer has a "cheat sheet" of RE syntax and common expressions.

The Watch, Modified, and Breakpoints windows will now show a tooltip on a mouse hover that contains a column's full text, if it is too wide to be shown entirely in the column.

If you "step out" (run to breakpoint or end) and you are in a CALL'd batch file, and if there are no more breakpoints in the current file, you will be returning to the parent batch file at the line following the CALL, and "step out" will be turned off.

CMDebug will not save a *.watch file if the only variables being watched are the default **?** and **_?**.

TCC-RT:

TCC-RT is compatible with the new Windows 10 Terminal (currently in preview).

Changed some of the less-commonly used dll's to load on demand, which will reduce the startup time and RAM footprint slightly.

All of the IPWorks internet / network / zip libraries have been updated.

The Onigmo regular expression library has been updated.

Added support for Python 3.8.

The [ ] wildcard now accepts either **!** or **^** as the NOT symbol.

The history and directory history popup windows now support multiple selection (with the shift or ctrl keys + left mouse), and they have a popup context menu (right mouse button) to Copy, Copy+Append, Cut, or Delete. You can also select multiple entries and execute them by pressing Enter - *TCC-RT* will create a command line that looks like this:

(*line1*) & (*line2*) & (*line3*)

There are new options for output redirection. These options will override the UnicodeOutput and UTF8Output directives in TCMD.INI. The piped output options also work with DOS pipes (i.e., |!:u). Note: these options only work for redirecting output from TCC internal commands.

>:a	Redirected output (STDOUT and/or STDERR) is ANSI (8 bit characters)
>:u	Redirected output is UTF16 Unicode
>:8 or >:u8	Redirected output is UTF8
>>:a	Appended redirected output (STDOUT and/or STDERR) is ANSI (8 bit characters)
>>:u	Appended redirected output is UTF16 Unicode
>>:8 or >>:u8	Appended redirected output is UTF8
|:a	Piped output is ANSI
|:u	Piped output is UTF16 Unicode
|:8 or l:u8	Piped output is UTF8

Redirection to CLIP: now defaults to UTF16.

Date and time ranges can now compare UTC times by adding a 'U' after the D or T (and the optional A, C, or W) in the range specification. For example:

/[twu00:00,11:59]

Size ranges now can test for compressed size (on NTFS drives with compression enabled for the file or directory) by appending a C to the S argument. For example, to specify files with a compressed size between 100 and 1000 bytes:

/[sc100,1000]

TCC-RT will detect if it is running as a service or detached before prompting for SSL or SSH authentication, and will provide an automatic 'Y' (yes) input.

<u>**Help:**</u>

The help is built with a new version of Help & Manual 7.

<u>**New TCC-RT Internal Variables:**</u>

There are a number of new internal variables for GPS position and status. They require a GPS (internal or external) that supports the Windows Sensor Location APIs. Note that your GPS may not support all the variables; if a value is not supported Windows will return "Ëlement not found". Unless otherwise specified, the values returned are decimal numbers (for example, 38.735122).

%_gpsalt - Altitude (from sea level) in meters.

%_gpsazimuth - Azimuth of each satellite in view. Returns a space-delimited list.

%_gpselevation - Elevation of each satellite in view. Returns a space-delimited list.

%_gpserrorradius - Accuracy of latitude and longitude values, in meters.

%_gpsfixquality - Quality of the fix (integer).

%_gpsfixtype - Type of the fix (integer).

%_gpshdop - Horizontal dilution of precision.

%_gpsheading - True heading.

%_gpsids - IDs of the satellites in view. Returned a space-delimited list of integers.

%_gpslat - Latitude

%_gpslon - Longitude

%_gpsmagheading - Magnetic heading.

%_gpsnmea - Returns the NMEA sentence as a string.

%_gpsopmode - GPS operation mode (integer).

%_gpspdop - Position dilution of precision.

%_gpsprns - PRN numbers of satellites in view. Returns a space-delimited list of integers.

%_gpssatsinview - Number of satellites in view (integer)

%_gpssatsused - Number of satellite used in solution (integer).

%_gpsselmode - GPS selection mode (integer).

%_gpssnr - Signal to noise ratio of each satellite in view. Returns a space-delimited list.

%_gpsspeed - Speed in knots.

%_gpsstatus - GPS status (integer).

%_gpsvdop - Vertical dilution of precision.

New TCC-RT Variable Functions:

@VARTYPE - Returns the type (if any) for the specified variable name. The possible values are:

0   No type
1   Integer (0-9)
2   Decimal (0-9, the decimal character, and the thousands separator)
3   Hex (0-9, A-F)
4   Boolean (0 or 1)
5   Alphabetic (A-Z and a-z)
6   Alphanumeric (A-Z, a-z, and 0-9)
7   Regular expression

@WINPATH - Convert from WSL pathname format to Windows format. For example:

echo %@winpath[//mnt/c/windows/system32/notepad.exe]
c:\windows\system32\notepad.exe

@WSLPATH - Convert from Windows pathname format to WSL format. For example:

echo %@wslpath[c:\windows\system32\notepad.exe]
//mnt/c/windows/system32/notepad.exe

Updated TCC-RT Variable Functions:

@EVAL - added log2() function.

@PID - added an optional second argument that specifies whether to return all PID's that match the first argument. For example:

@pid[firefox,+]

@TIMER - Now uses the Windows performance counters for higher resolution. The default @TIMER resolution is in milliseconds (.001 seconds) instead of hundredths (0.01 seconds). @TIMER has three new values for the optional second argument to return the split time as an arithmetic value:

ms - split time in milliseconds
us - split time in microseconds
ns - split time in nanoseconds

@VERSION - added a new optional 5th parameter that specifies whether to append the version number to the filename (0), or prefix it to the extension (1).

Updated Commands:

ACTIVATE

/POS - accepts a * value for any of the arguments. If the value is *, ACTIVATE will use the existing position / width / height value. For example, to resize a window without moving it:

ACTIVATE "title" /POS=*,*,1200,800

To move a window without resizing it:

ACTIVATE "title" /POS=200,400,*,*

ASSOCIATE

/Pn - the /P(ause) option has an optional argument *n* that specifies the number of seconds to wait for a keystroke before continuing.

/V:verb- ASSOCIATE defaults to reading and writing to SHELL\OPEN\COMMAND. You can use a different verb by specifying the /V option. For example, to tell create a PRINT verb for .TXT files:

ASSOCIATE /V:PRINT .txt=%%SystemRoot%%\system32\NOTEPAD.EXE /p %%1

ATTRIB

/Pn - the /P(ause) option has an optional argument *n* that specifies the number of seconds to wait for a keystroke before continuing.

CHCP

/Pn - the /P(ause) option has an optional argument *n* that specifies the number of seconds to wait for a keystroke before continuing.

COLOR

COLOR now supports changing the console color palette with either an .INI file (for example, as used by the ColorTool utility), or an .ITERMCOLORS file. The syntax is:

COLOR /F filename

If you are running in a Take Command tab window, COLOR will pass the new colors to TCMD to update the tab window. You can have a different color palette in each tab window.

DIR

Directory colorization (using either the COLORDIR environment variable or OPTION / Colors / Directory colors) now supports all types of ranges (size, date, time, description, owner, and exclusion). The syntax is the same as for ranges in an internal command. For example, to display files that are between 100 and 1000 bytes in bright green:

set colordir=/[s100,1000]:bri green;

Directory colorization now supports colors for file subsystem types. The supported subsystems are:

EXETYPE_WIN32GUI	Windows x86 GUI app
EXETYPE_WIN32CUI	Windows x86 console app
EXETYPE_WIN64GUI	Windows x64 GUI app
EXETYPE_WIN64GUI	Windows x64 console app
EXETYPE_DOS	DOS (16-bit) app (obsolete)
EXETYPE_POSIX	POSIX app (obsolete)
EXETYPE_EFI	EFI app

For example, to display 32-bit console apps in bright green and 64-bit console apps in bright red:

set colordir=EXETYPE_WIN32CUI:bri green;EXETYPE_WIN64CUI:bri red

/-C - Removes the thousands separators when displaying file sizes (for compatibility with CMD.EXE).

/Pn - the /P(ause) option has an optional argument *n* that specifies the number of seconds to wait for a keystroke before continuing.

ENUMSERVERS

/Pn - the /P(ause) option has an optional argument *n* that specifies the number of seconds to wait for a keystroke before continuing.

ENUMSHARES

/Pn - the /P(ause) option has an optional argument *n* that specifies the number of seconds to wait for a keystroke before continuing.

FOR

The ~a (display attributes) format has been updated to match the current CMD behavior with extended attributes (including CMD's behavior of not displaying all the extended attributes).

GOSUB

GOSUB now supports calling subroutines in another file when that file is compressed.

HEAD

/Pn - the /P(ause) option has an optional argument *n* that specifies the number of seconds to wait for a keystroke before continuing.

IF

ISVISIBLE "title" - executes the command if the specified window is visible. (This means that Windows has set the visibility flag; it does not mean that the window is necessarily visible on your desktop.

IFF

ISVISIBLE "title" - executes the command if the specified window is visible.

INSTALLED

/Pn - the /P(ause) option has an optional argument *n* that specifies the number of seconds to wait for a keystroke before continuing.

LIBRARY

/Pn - the /P(ause) option has an optional argument *n* that specifies the number of seconds to wait for a keystroke before continuing.

PAUSE

/C - After you press a key, erases the prompt and does not print a CR/LF.

PDIR

Directory colorization now supports ranges (see DIR for details).

Directory colorization now supports subsystem types (see DIR for details).

/D - switched the meaning from "colorize" to "don't colorize" (to match DIR and SELECT).

/Pn - the /P(ause) option has an optional argument *n* that specifies the number of seconds to wait for a keystroke before continuing.

PLUGIN

/Pn - the /P(ause) option has an optional argument *n* that specifies the number of seconds to wait for a keystroke before continuing.

PRINT

/S *printer* - set the default printer.

PRIORITY

If you only provide a PID or window title, PRIORITY will display the current priority.

/Pn - the /P(ause) option has an optional argument *n* that specifies the number of seconds to wait for a keystroke before continuing.

REGDIR

/Pn - the /P(ause) option has an optional argument *n* that specifies the number of seconds to wait for a keystroke before continuing.

/X - display the REG_DWORD, REG_DWORD_BIG_ENDIAN, and REG_QWORD values in hex. Only valid when used with /V and /D.

SELECT

Directory colorization now supports ranges (see DIR for details).

Directory colorization now supports subsystem types (see DIR for details).

SERVICES

/I - Display the PID's for services. Note that stopped services will return 0 for the PID, as will Windows services.

SYNC

/WAIT=n - Pause for n milliseconds between each block copied from the source to the target file. This is useful for slow networks and very large file copies; it prevents SYNC from monopolizing all of the network I/O.

TAIL

/Pn - the /P(ause) option has an optional argument *n* that specifies the number of seconds to wait for a keystroke before continuing.

TASKLIST

/Pn - the /P(ause) option has an optional argument *n* that specifies the number of seconds to wait for a keystroke before continuing.

TIMER

TIMER now uses the Windows performance counters for greater accuracy. The default TIMER resolution is now in milliseconds (.001 seconds) instead of hundredths (0.01 seconds).

/L - When used with /S (split time) or TIMER OFF, display the result in the number of milliseconds.

/M - When used with /S (split time) or TIMER OFF, display the result in the number of microseconds.

/N - When used with /S (split time) or TIMER OFF, display the result in the number of nanoseconds.

TOUCH

/CD - create the specified directory if it doesn't exist.

TREE

/Pn - the /P(ause) option has an optional argument *n* that specifies the number of seconds to wait for a keystroke before continuing.

TYPE

/Pn - the /P(ause) option has an optional argument *n* that specifies the number of seconds to wait for a keystroke before continuing.

UNQLITE

Updated to UnQLite 1.1.9.

WINDOW

/POS - accepts a * value for any of the arguments. If the value is *, WINDOW will use the existing position / width / height value. For example, to resize a window without moving it:

    WINDOW /POS=*,*,1200,800

To move a window without resizing it:

    WINDOW /POS=200,400,*,*

2.8   What's New in Version 24

CMDebug 24.02:

Updated the internet and compression libraries in TCC-RT.

Changed the TCC-RT /S startup switch to /B.

Added support for the CMD /S startup option to TCC-RT.

Updated the Language dlls.

Updated the Scintilla (scilexer.dll) editor.

CMDebug 24.01:

The Take Command display output is faster and uses less CPU.

TCC now supports Python 3.7.2.

Improved the editor performance in IDE, CMDebug and TCEdit.

Everything Search has been updated to version 1.4.1.932.

Updated the Onigmo regular expression library.

Updated the Scintilla (scilexer.dll) editor.

CMDebug 24.0:

Installer:

CMDebug is using new version of Advanced Installer.

CMDebug:

CMDebug is compatible with the Windows 10 Fall 2018 Update.

Version 24 is using a new version of the GUI framework library.

Version 24 is using a new version of the Scintilla editor.

Text display is faster and the antialiased fonts are a little clearer.

The dark themes now have dark margins (profiler, breakpoints, line numbering).

When you move the **CMDebug** window, it will snap to the screen edges if it's within 10 pixels.

CMDebug includes TCC-RT 24.0.

Help:

The v24 help is built with a new version of the help compiler (Help & Manual).

The help has been expanded with more examples and key words.

Updated TCC-RT Commands:

COPY

/Nz - Skip system directories (when used with /S).

DEDUPE

/Nz - Skip system directories (when used with /S).

DEL

/Nz - Skip system directories (when used with /S).

DELAY

DELAY UNTIL now accepts a space, comma, or = between the date and the time. (This allows it to work with the string returned by @AGEDATE.)

DIR

/Nz - Skip system directories (when used with /S).

LIBRARY

/R now supports reading multiple library files.

/Q - Don't display an error if the function doesn't exist.

MOVE

/Nz - Skip system directories (when used with /S).

PATH

/V - Checks all of the directories in %PATH, and displays an error message for any that don't exist.

New TCC-RT Commands:

UNLIBRARY

UNLIBRARY removes library functions defined with the LIBRARY command. The syntax is:

UNLIBRARY [/Q /R filename ... (function ...)] functionname ...

/Q - Don't display errors if the library function doesn't exist.

/R - Read the functions to delete from a file

UNLIBRARY supports exception lists (enclosed in parentheses) to specify library functions you do not want to delete.

2.9 What's New in Version 23

CMDebug 23.0:

Installer:

CMDebug is using new version of Advanced Installer.

CMDebug:

CMDebug is compatible with the Windows 10 Spring 2018 Update.

CMDebug is using an updated version of the GUI framework.

CMDebug is using an updated version of the Scintilla edit control.

CMDebug is using an updated version of the Onigmo regular expression library.

CMDebug has a number of controls drawing improvements related to custom font sizes and DPI scaling.

The Tabs context menu (right click on a tab header) has three new options:

> Open Containing Folder (opens a File Explorer window in the batch file's directory)
> Close
> Close All But This

If you are editing a variable name in the Watch window, the Delete (X) button will delete marked text in the edit control. Otherwise, X will delete the currently selected line in the watch list.

"Run to Cursor" is a new option in the Debug menu. If you click on a line in the debugger window, and then select "Run to Cursor", the debugger will run the batch file (ignoring any breakpoints) until it reaches the selected line.

CMDebug includes TCC-RT 23.0.

Help:

The v23 help is built with a new version of the help compiler (Help & Manual).

The help has been expanded with more examples and key words.

2.10   What's New in Version 22

CMDebug 22.0:

Installer:

CMDebug is using new version of Advanced Installer.

CMDebug:

We have made additional changes to *CMDebug* to make it harder to attack with malware.

There are hundreds of minor tweaks to the layout, icons, menus, and themes (particularly the dark themes).

CMDebug is compatible with the Windows 10 Fall Creators Update.

CMDebug is using an updated version of the GUI framework.

CMDebug is using an updated version of the Scintilla edit control.

CMDebug is using an updated version of the Onigmo regular expression library.

There is a new Tools menu with three commands:

- Regular Expression Analyzer
- Lookup Windows Errors
- Character Map

Added autocomplete for the edit controls and combo boxes that take (existing) file and directory names.

If launched from *CMDebug* (Tools / View Errors), the Lookuperrors.exe app will center itself in the CMDebug window.

The Regular Expression Analyzer now lets you select the regular expression syntax you want to test (Perl, Python, Ruby, Gnu, etc.).

Added Python and Emacs syntax options to regular expressions.

The tab edit windows now support screen readers.

CMDebug tabs and toolbars now use ClearType for cleaner text.

CMDebug tabs now display the tab text in bold when selected.

The debugger will automatically save & reload watch lists (*.watch).

The debugger will automatically save & reload bookmarks (*.bmark).

The debugger will automatically save & reload breakpoints (*.bp).

The debugger edit windows now support screen readers.

The debugger tabs now use ClearType for cleaner text.

The debugger tabs now display the tab text in bold when selected.

Help:

The v22 help is built with a new version of the help compiler (Help & Manual).

3   CMDebug IDE / Batch Debugger

CMDebug is a very powerful IDE (Integrated Development Environment) for creating, editing, and debugging batch files. The IDE includes syntax coloring for batch files (.BAT, .CMD, and (for *TCC-RT*) .BTM) and code folding for command groups and the *TCC-RT* DO, IFF, SWITCH, and TEXT commands.

If you are creating or debugging a *TCC-RT* batch file, use the **TCC Syntax** in the Options menu. If you are creating or debugging a batch file to run under CMD.EXE, select the (default) **CMD Syntax**. If you select **CMD Syntax,** the debugger will reconfigure the batch file parser for maximum **CMD.EXE** compatibility, including disabling *TCC-RT-*only internal commands, aliases, variables, functions, and plugins.

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.)

CMDebug sets the environment variable BATCH_DEBUGGING=1. A batch file can test for that variable if it needs to know when it's being debugged.
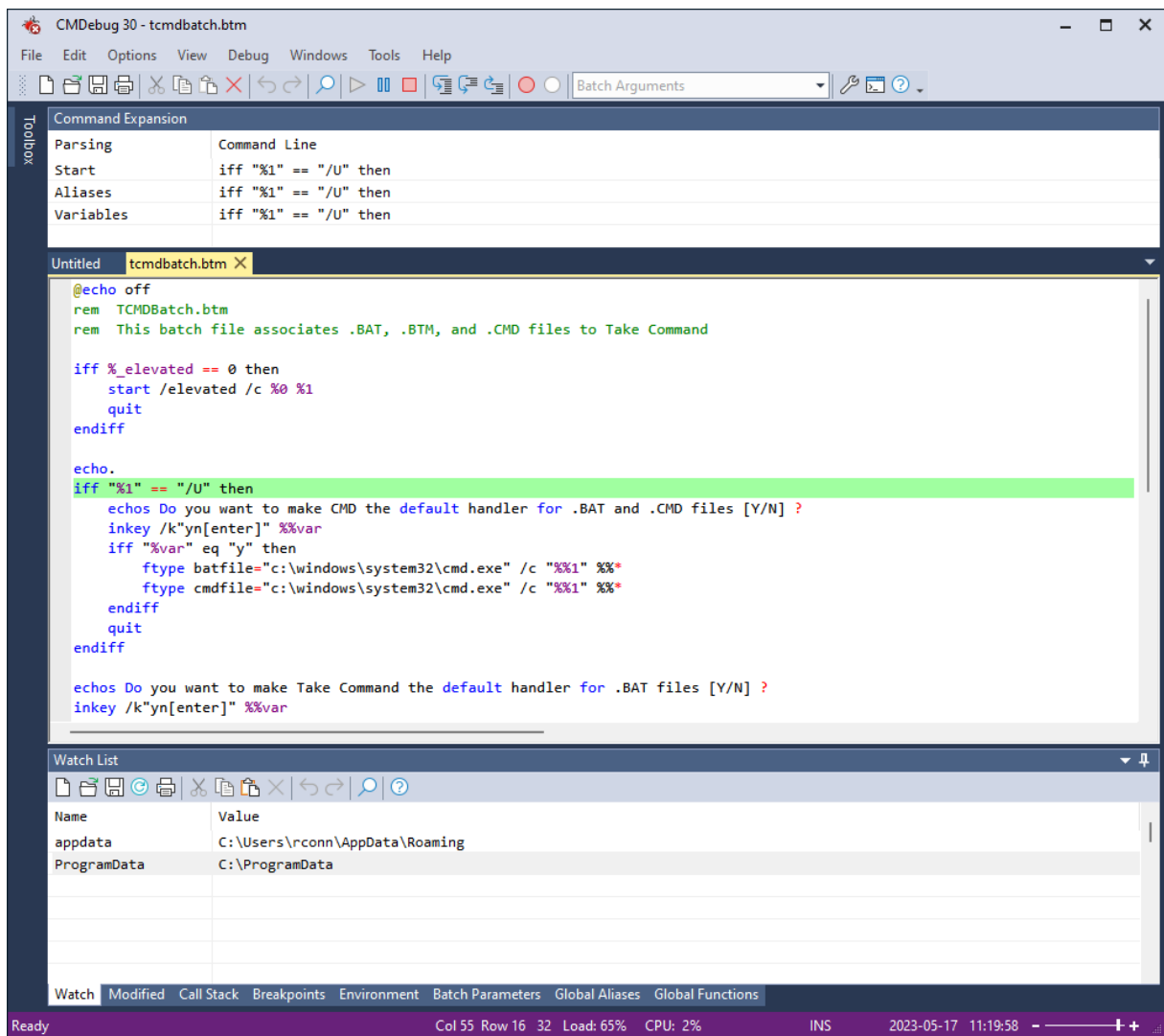
3.1 Installing CMDebug

You can download the latest version of *CMDebug* from our website at:

https://jpsoft.com/all-downloads/downloads.html

To install *CMDebug*, run the downloaded self-extracting installer (tcmd.exe). *CMDebug* uses the Windows Installer, so the installation options will be the same as most other Windows applications.

TCC-RT comes in both 32-bit (x86) and 64-bit (x64) versions. The *CMDebug* installer is 64-bit only (32-bit versions are available upon request for multisystem licenses).

3.2 Registration

There are no separate **trial** and **registered** versions of *CMDebug.* 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. You can view your registration info with the Help/About menu entry in *CMDebug*.

When you purchase a new or upgrade copy of *CMDebug*, you will receive an email with your name and registration key. Start *CMDebug*, click on the **Help** menu entry and then **Register**. 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 operations@jpsoft.com.

If you need to remove your *CMDebug* registration from a computer, click on the **Help** menu entry and then **Register**. Enter the activation key you used to register *CMDebug* and then click on the **Unregister** button.

3.3 Starting CMDebug

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

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

When you configure a *CMDebug* 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 *CMDebug.* For example:

```
Command Line:          C:\Program Files\JPSoft\CMDebug 25\CMDEBUG.EXE
Working directory:     C:\
```

You do not need to use the Change Icon button, because **CMDEBUG.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 **CMDebug** startup command line does not need to contain any information. However, you may add information to the startup command line that will affect the way **CMDebug** operates.

CMDebug Startup Options

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

```
"c:\program files\jpsoft\cmdebug21\cmdebug.exe"
```

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

```
d:\path\cmdebug.exe [[/]@d:\path\inifile] [//directive=value...] [/D d:
    \path] [/N /X] [/C command] [/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 **CMDebug** path and executable name (**CMDEBUG.EXE**):

```
d:\path\cmdebug.exe
```

TCC-RT Syntax

If you are creating & debugging **TCC-RT** batch files, there are some additional options for the **CMDebug** command line.

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. You don't need this option if:

1)   your .INI file is named *TCMD.INI,* and
2)   it is in one of the following directories:
    2.1)   the same directory as **CMDebug**
    2.2)   the "%programdata%\JP Software\CMDebug 21" directory
    2.3)   the %localappdata% directory

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

To start **CMDebug** without any .INI file, you can create an empty file and specify it as your .INI file.

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

CMDebug will skip the leading forward slash.

Options:

//directive=value
> This option tells **CMDebug** 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, 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.

/A
> This option causes the output of internal commands to a pipe or redirected to a file to be in ASCII when **CMDebug** starts. This is the default value, and isn't necessary unless you want to override a Unicode Output configuration option.

/C
> If the specified batch file doesn't exist, create it without prompting.

/D
> Disable execution of AutoRun commands from Registry. If /D is not specified when **CMDebug** 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

/I
> Don't load the .INI file, execute TCSTART or TCEXIT, or load plugins. You can optionally specify any combination of individual arguments:

> | | |
> |---|---|
> | **/II** | Don't load the .INI file. |
> | **/IL** | Don't load the default library functions. |
> | **/IP** | Don't load plugins. |
> | **/IS** | Don't execute TCSTART. |
> | **/IX** | Don't execute TCEXIT. |

> For example, /IILPSX will disable everything.

/L:
> Forces the use of local lists as the default 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 and FUNCTION for more details.

> You can optionally specify individual arguments:

> | | |
> |---|---|
> | **/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. |

/S	Disable ^C and ^Break.

/T:*bf*   This option sets the foreground and background colors in the **CMDebug** 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*.* See the CMD color codes in Colors, Color Names & Codes.

In most cases you should set default colors with the corresponding Output Colors 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-RT starts. The command :

        OPTION //UnicodeOutput=yes | no

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

/U8   This option causes the output of internal commands to a pipe or redirected to a file to be in UTF-8 when TCC-RT starts. The command :

        OPTION //UTF8Output=yes | no

may be used at any time to switch between UTF-8 and ASCII output.

/V   Tells **CMDebug** to handle the CMD syntax ***!varname!*** as a delayed expansion of ***%varname***. Since CMD, unlike **TCC** or **TCC-RT**, 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 **CMDebug** to alter the operation of the MD (MKDIR) command to automatically create all necessary intermediate directories when it creates a new subdirectory. Its effect is the same as adding a /S option to all MD (MKDIR) commands. This option is included for compatibility with CMD, where it also enables other options. However, in **CMDebug** those options are already enabled by default.

3.4   Console Window

CMDebug will create a console window when it starts. All batch input and output will be done in the batch window, so you can see exactly how the batch file will run when executed directly from Windows or the command processor.

3.5   CMDebug Menus

- File
- Edit
- Options
- View
- Debug
- Windows
- Help

3.5.1 File

The **File** menu allows you to create new or open existing batch files, save or print the edit windows, or exit *CMDebug*.

New

Opens a new (empty) edit window.

Open

Open the specified file in a new edit window. If the file is a batch file, **Open** will load the batch file and any associated breakpoint file (filename.ext.bp) and the watched variables file (filename.ext.watch).

Reload from Disk

Reloads the file in the current tab from the disk.

Close

Close the current edit window. If the file has been modified, you will be prompted to save it.

Close All

Close all edit windows. If the files have been modified, you will be prompted to save them.

Save

Saves the contents of the current edit window. A *Save As* dialog box appears in which you can enter the name of the file that you wish to use. If the file is a batch file, **Save** will save the batch file and any associated breakpoint file (filename.ext.bp) and the watched variables file (filename.ext.watch).

Save As

Saves the contents of the current edit window. A *Save As* dialog box appears in which you can enter the name of the file that you wish to use.

Save Copy As

Saves the contents of the current edit window. A *Save As* dialog box appears in which you can enter the name of the file that you wish to use.

Save All

Saves the contents of the current edit window to a file. A *Save As* dialog box appears in which you can enter the name of the file that you wish to use.

Encoding

Files are always treated as UTF-8 inside the editor. This option allows you to specify how the file will be written when it is saved to disk.

Default Codepage   When the file is saved it will be written using the current codepage

UTF16 Little Endian	When the file is saved it will be written as UTF-16
UTF8	When the file is saved it will be written as UTF-8
UTF8 with BOM	When the file is saved it will be written as UTF-8 with a leading BOM

Move to Recycle Bin

Delete the file in the current tab by sending it to the Recycle Bin.

Print...

Sends the contents of the current edit window to the printer. A Print dialog box appears in which you can choose the portion of the screen buffer you wish to print.

Print Preview

Show a Print Preview window for the current edit window.

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.

File Properties

Display the Windows properties dialog for the file in the current edit window.

Load Session...

Open a session file previously saved with the **"Save Session"** option (see below), which loads the specified files into tab windows.

Save Session...

Create a *.session* file with the names of the files in the tab windows. A subsequent "Load Session" will open the files in the *.session file you specify.

Exit

Ends the current CMDebug session.

3.5.2   Edit

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.

Undo

Undo the last action in the active editor window.

Redo

Redo the last action in the active editor window.

Cut

Copies selected text to the clipboard and deletes it from the editor.

Copy

Copies selected text to the clipboard.

Copy+Append

Append the current selection to the existing clipboard content.

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. Paste will check to see if the Ctrl + Shift keys are down, and if so it will insert a " & " between lines of a multiline paste.

Delete

Deletes the selected text from the editor.

Duplicate

Copy the existing line and insert it before the current line.

Select All

Marks the entire contents of the active editor window as selected text.

Move Line Up

Move the selected (highlighted) line up one row.

Move Line Down

Move the selected (highlighted) line down one row.End f

Tabs...

Displays a dialog to set the tab width, indentation width, and whether to insert tabs as spaces and to reformat indentation.

End of Line Characters

CR + LF   Lines end in a Carriage Return + Linefeed (Windows default)
CR   Lines end in a Carriage Return (OSX default)
LF   Lines end in a Linefeed (Linux default)

Convert End of Line Characters

Convert the end of line character(s) in the current tab window to the character(s) defined in the "End of Line Characters" menu entry above.

Insert Folder

Displays the Windows folder selection dialog and puts the selected directory name at the current position in the editor.

Insert Filename...

Displays the Windows file selection dialog and puts the selected filename at the current position in the editor.

Find...

Search the contents of the editor window using [regular expressions](#).

Replace...

Search and replace text in the editor window.

Goto...

Move the cursor to the specified line number.

Advanced

Toggle Comment

Inserts or removes a **rem** statement at the beginning of the current command line.

Remove Blank Lines

Remove blank lines from the file in the active edit window.

Compress Spaces

Remove extra spaces from the file in the active edit window.

Tabify Selection

Replace spaces with tabs in the selected text in the active edit window.

Untabify Selection

Replace tabs with spaces in the selected text in the active edit window.

Make Selection Upper Case

Convert the selected text to upper case.

Make Selection Lower Case

Convert the selected text to lower case.

View Whitespace

Shows a dot for space & tab characters.

View EOL

Show the CR & LF characters.

Toggle Current Fold

Toggles code folding for the current line.

Toggle All Folds

Toggles code folding for the file in the active edit window. When using **CMD syntax**, this only affects command groups. When using **TCC syntax**, this also affects IF, DO, and SELECT.

Bookmarks

Toggle Bookmark

Create or remove a bookmark on the current line in the active edit window.

Next Bookmark

Go to the next bookmark in the active edit window.

Previous Bookmark

Go to the previous bookmark in the active edit window.

Clear All Bookmarks

Removes all bookmarks from the active edit window.

The debugger will automatically save bookmarks (the current batch file name + ".bmark"), and reload them the next time the batch file is loaded in the debugger.

Add to Watch

Add the selected variable to the Watch window.

Evaluate Selection

Call the "Evaluate Expression" dialog and evaluate the highlighted expression.

Read Only

Change the current edit window to read-only, disabling all modify / write operations.

3.5.3   Options

TCC Syntax

If enabled, *CMDebug* will use the TCC syntax colorizer, and enable the extended *TCC-RT* commands, variables, and functions. This is the default for .BTM files.

CMD Syntax

If enabled, *CMDebug* will use the CMD syntax colorizer, and disable the extended *TCC-RT* commands, variables, and functions. This is the default for .BAT and .CMD files.

Syntax Colors...

Select the colors used in the syntax colorization from a 16 million color palette. When you click on one of the foreground or background color buttons, *CMDebug* will display a color picker dialog to let you choose colors.

Font

Displays a font selection dialog. The font will be used in the edit windows and the Watch, Modified, Breakpoints, Environment, Batch Parameters, Aliases, and Functions tab windows.

Caret Color...

Set the caret color for the tab edit windows.

Profiler

Toggles the batch file profiler timer on and off. When the Profiler is on, it will display the elapsed time for each command line in the margin immediately to the left of the command line.

Display Line Numbers

Display line numbers in the left margin.

Display Folding Margin

Toggles the code folding margin (the + indicator) on and off. Code folding supports command groups and (*TCC-RT* only) DO, IFF, SWITCH, and TEXT.

Indentation Guides

Toggles the vertical lines at the current indent columns (this is useful for lining up code).

Always on Top

If enabled, forces the *CMDebug* window to be the top-most (i.e., always on top of other windows).

Themes

Select a predefined Visual Studio-style theme for *CMDebug*. This will change the color and appearance of the *CMDebug* windows and borders.

VS 2005
VS 2008
VS 2010
VS 2012
VS 2012 Dark
VS 2015
VS 2015 Dark
VS 2015 Blue

Window Tabs...

Select the window tab location (top, bottom, left, or right).

3.5.4   View

Toggle current fold

Toggles folding the current line on & off.

Toggle all folds

Toggles every fold in the file.

Toolbar

Show or hide the *CMDebug* toolbar.

Status Bar

Show or hide the status bar.

Toolbox

The Toolbox window lists all of the internal commands, variables, and variable functions organized by category. Selecting a command or variable and then pressing F1 will display the help for that command/variable.

Command Expansion

The "Command Expansion" window will pop up above the tab window when you start debugging. The Command Expansion window shows the original command line, the command line after alias expansion, and the command line after variable expansion. This is a docking window, so it can be moved & attached at other locations. If you don't want to see the Command Expansion window, you can turn it off from the IDE "View / Command Expansion" menu option.

Watch

The Watch window allows you to monitor variables.

When using **TCC syntax**, the Watch tab defaults to showing two variables:

%_? - The last TCC-RT result value
%? - The last ERRORLEVEL value

Open will load the batch file and any associated breakpoint file (filename.ext.bp) and the watched variables file (filename.ext.watch).

Save will save the batch file and any associated breakpoint file (filename.ext.bp) and the watched variables file (filename.ext.watch).

If you right click in the first column of the Watch window, the debugger will pop up an environment variable listbox. If you select an entry, it will be added to the watch list.

The watch window now also supports internal variables, variable functions, and user-defined functions.

Modified

The Modified tab window shows all variables that are created or modified while executing the batch file. (This is like the "Auto" window in Visual Studio.)

Breakpoints

The Breakpoints tab window shows the breakpoints for the active tab window. (Line number, Count, and optional condition.)

Environment

Show the Windows environment *CMDebug* is using when it runs batch files. You can add, modify, or delete variables.

Batch Parameters

Display the batch file arguments (%0 - %n). You can modify the batch arguments while debugging the batch file. To specify batch arguments when the batch file starts, you can enter them in the Batch Arguments combobox on the toolbar.

Call Stack

Display the current call stack (the filename, line #, command line, and the line(s) that called it (i.e., GOSUB or CALL). Double-clicking on a line in the Call Stack window will take you to that line in the tab window. (Note that the call stack is only expanded when you "Step Into" the next command.)

Local Aliases
Global Aliases

(*TCC-RT syntax* only) Show the user-defined functions. You can add, modify, or delete aliases in the edit window. If you are using CMD Syntax, this window will be disabled.

Local Functions
Global Functions

(*TCC-RT syntax* only) Show the user-defined functions. You can add, modify, or delete functions in the edit window. If you are using CMD Syntax, this window will be disabled.

Command Prompt

Start a new command prompt window.

3.5.5   Debug

Start

Start debugging a .BAT, CMD, or .BTM batch file

Start Without Debugging

Run a .BAT, CMD, or .BTM batch file (no debugging)

Pause

Pause debugging for the batch file

Stop Debugging

Stop debugging the batch file

Show Next Statement

Show the next statement

Step Into

Step into the next statement

Step Over

Step over

Run to Breakpoint or End

Run until the debugger reaches a breakpoint. If you are in a CALL'd batch file, and there are no more breakpoints in the current file, you will be returned to the parent batch file at the line following the CALL, and "Run to Breakpoint or End" will be turned off.

Skip This Line

Skip this line and continue execution with the next line

Run to Cursor

If you click on a line in the debugger window and select "Run to Cursor", the debugger will execute the batch file (ignoring any breakpoints) until it reaches the selected line.

Jump to This Line

If you click on a line in the debugger window and select "Jump to This Line", the debugger will continue execution starting with the selected line.

Pause on Error

Pause if the debugger encounters a Windows or internal command error

Toggle Breakpoint

Set or clear a breakpoint on the current line

Next Breakpoint

Go to the next breakpoint

Previous Breakpoint

Go to the previous breakpoint

Clear All Breakpoints

Clear all breakpoints

Enable All Breakpoints

Enable all breakpoints

Disable All Breakpoints

Disable all breakpoints

Error Lookup

Opens a small dialog that lets you look up Windows and network error messages based on the integer value.

Evaluate Command

Display a dialog and execute the command on the current line.

Evaluate Expression

Display a dialog and expand variables and aliases on the line

Evaluate Selection

Expand variables and aliases in the selected text

Add to Watch

Add the selected variable to the Watch window

3.5.6   Windows

Zoom In

Increase the tab window font size by one point.

Zoom Out

Decrease the tab window font size by one point.

Reset Zoom

Revert to the original editor window font size.
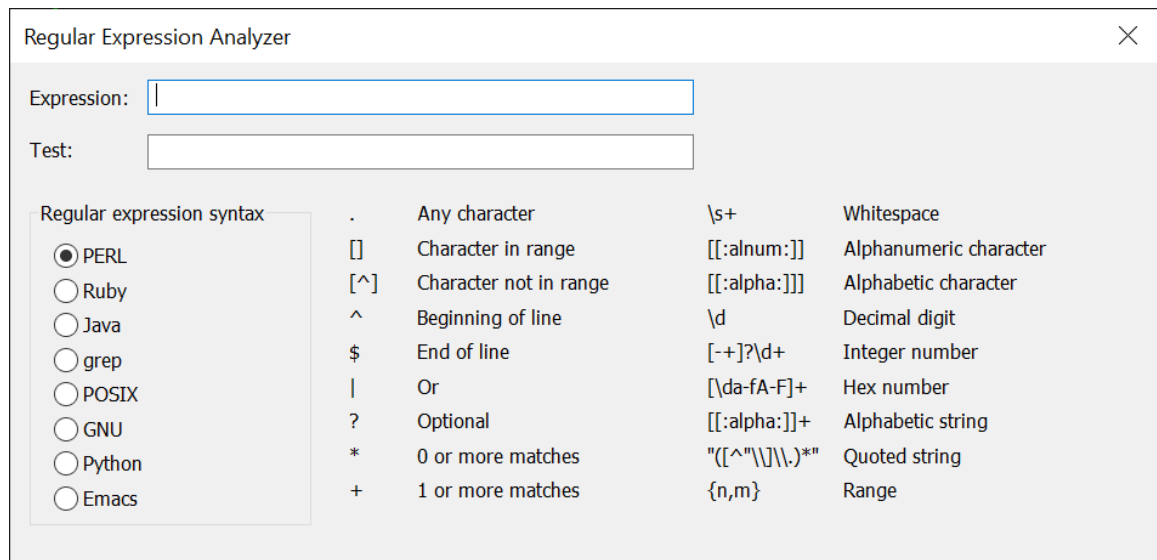
3.5.7   Tools

Regular Expressions...

CMDebug includes a regular expression analyzer dialog. There are two edit boxes:

1) The first is for the regular expression to test.. If the regular expression is valid, the dialog will display a green check to the right of the expression edit box. If the regular expression is invalid, the dialog will display a red X.
2) The second edit box is for the text you want to match against the regular expression. If the text matches the regex, the dialog will display a green check to the right of the test edit box. If the text doesn't match, the dialog will display a red X.

You can also choose the regular expression syntax you want to use (Perl, Ruby, Java, etc.). The analyzer will default to the current default for **Take Command**.

The analyzer has a microsecond timer (to the right of the "Test" edit control) that measures the time it took to evaluate the expression.

Regular Expression Analyzer   ✕

Expression: [   ]

Test: [   ]

Regular expression syntax					
	.	Any character	\s+	Whitespace
⦿ PERL	[]	Character in range	[[:alnum:]]	Alphanumeric character
○ Ruby	[^]	Character not in range	[[:alpha:]]]	Alphabetic character
○ Java	^	Beginning of line	\d	Decimal digit
○ grep	$	End of line	[-+]?\d+	Integer number
○ POSIX	|	Or	[\da-fA-F]+	Hex number
○ GNU	?	Optional	[[:alpha:]]+	Alphabetic string
○ Python	*	0 or more matches	"([^"\\]\\.)*"	Quoted string
○ Emacs	+	1 or more matches	{n,m}	Range

Lookup Errors...

Opens a small dialog that lets you look up Windows, network, and NSTATUS error messages based on the integer value. You can enter hex input with a leading **x** or **0x**.

❌ Windows Error Lookup   —   □   ✕

Error: [31]   [Find]   ☐ Always on Top

A device attached to the system is not functioning.
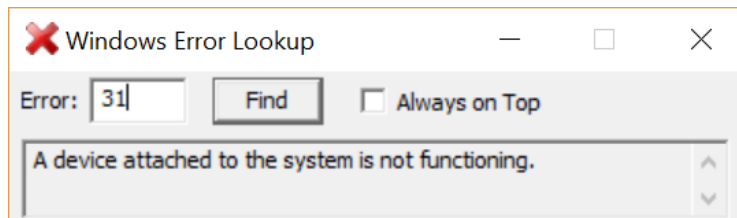
Character Map...

Opens the Windows Character Mapping dialog that lets you look up and copy characters for any font.
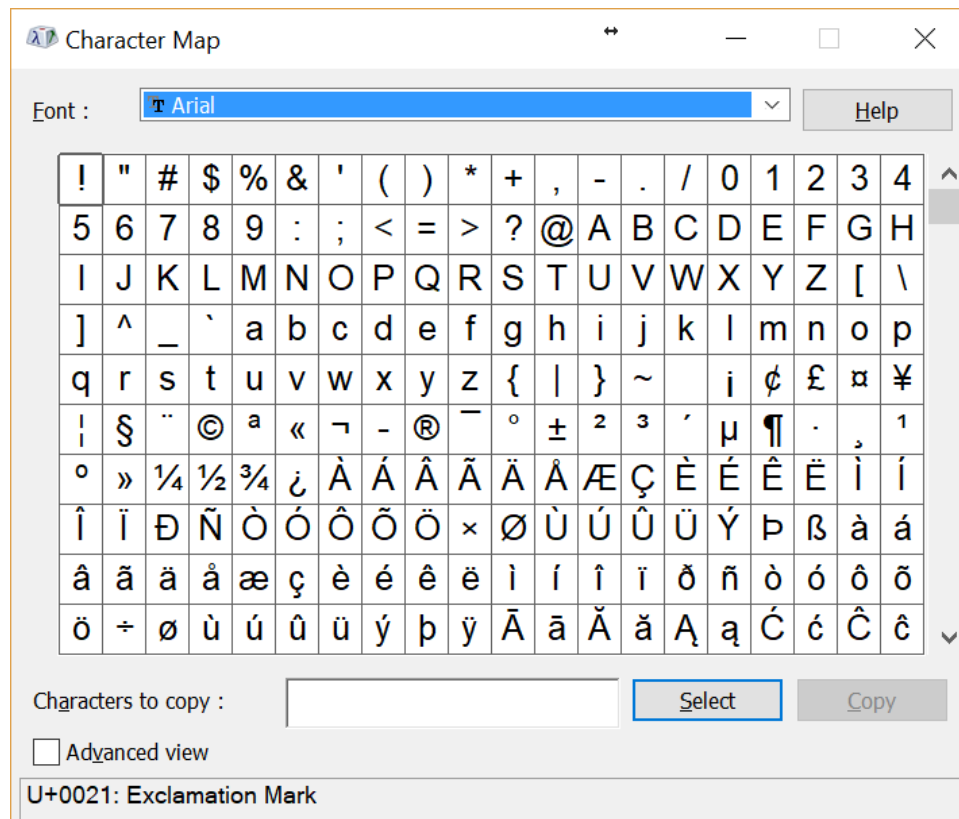
Command Prompt

Start a new **TCC** command prompt window.

3.5.8   Help

Help

Display the **CMDebug** help file (CMDHELP.CHM), from which you can directly navigate to any topic.

Dynamic Help

(**TCC-RT syntax only**) Display help for the command or variable at the current cursor position.

Search Topics

Displays the **Search** window of the **CMDebug** help file.

Index

Displays the **Index** window of the **CMDebug** help file.

https://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.

JP Software Forums

A hyperlink to the JP Software support forums.

Feedback

Leave a suggestion in the JP Software feedback forum.

Register

Enter an activation key to register *CMDebug*.

Check for Updates

Query the JP Software web server to see if there is an updated version of *CMDebug* 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.

About

Displays the *CMDebug* version, copyright, and license information.

3.6   Toolbar

The *CMDebug* toolbar has a number of icons to control editing and debugging. Each has a tooltip for quick reference:

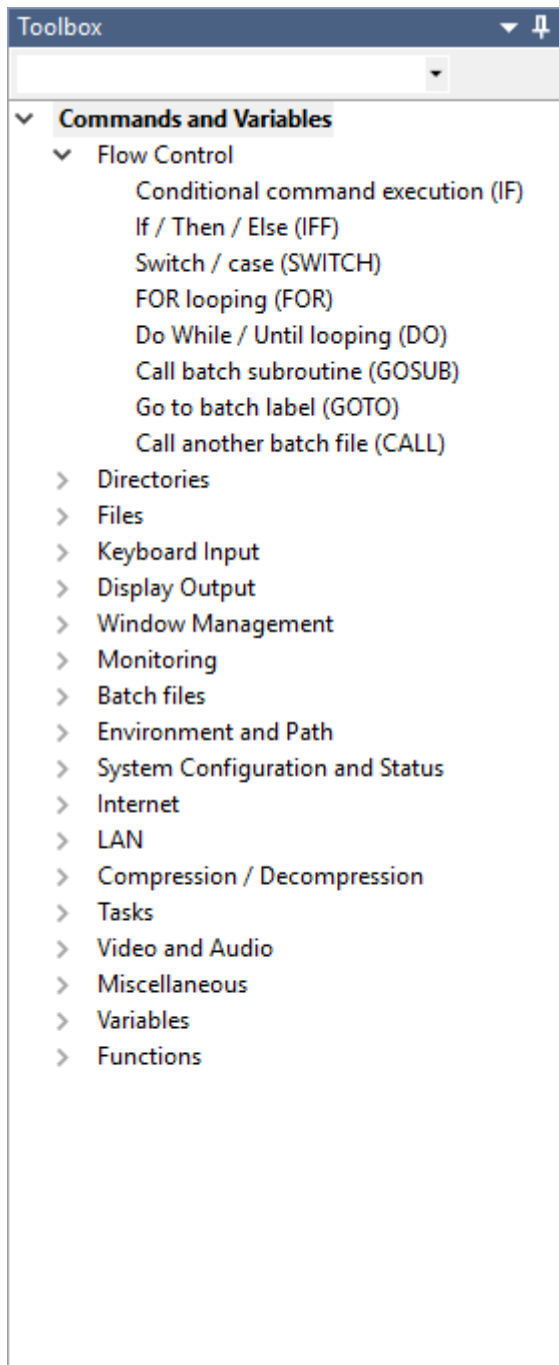|	
New	Create a new batch file in a new tab window.
Open	Open an existing batch file in a new tab window.
Save	Save the current batch file.
Print	Print the current batch file.
Cut	Copy the highlighted selection to the clipboard and delete it from the file.
Copy	Copy the highlighted selection to the clipboard.
Paste	Copy the contents of the clipboard to the current cursor location.
Delete	Delete the highlighted selection.
Undo	Undo the last edit.
Redo	Restore the last Undo.
Find	Search for text.
RegEx Filter	Hides all lines that don't match the regular expression,
Batch Arguments	New batch file arguments. The text will be parsed into %1 - %n batch arguments and used when the batch file is debugged.
Start Debugging	Starts the debugger. The cursor will be placed on the first line.
Pause Debugging	Pause execution at the next line.
Stop Debugging	Stops the debugger.

Step Into	Execute the current line.
Step Over	Execute the current line but disable the debugger during 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	Clear all breakpoints in the current batch file.
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.)
Help	Display the online help.

The *CMDebug* toolbar is customizable. To customize the toolbar click on the down arrow on the right side of the toolbar**.**
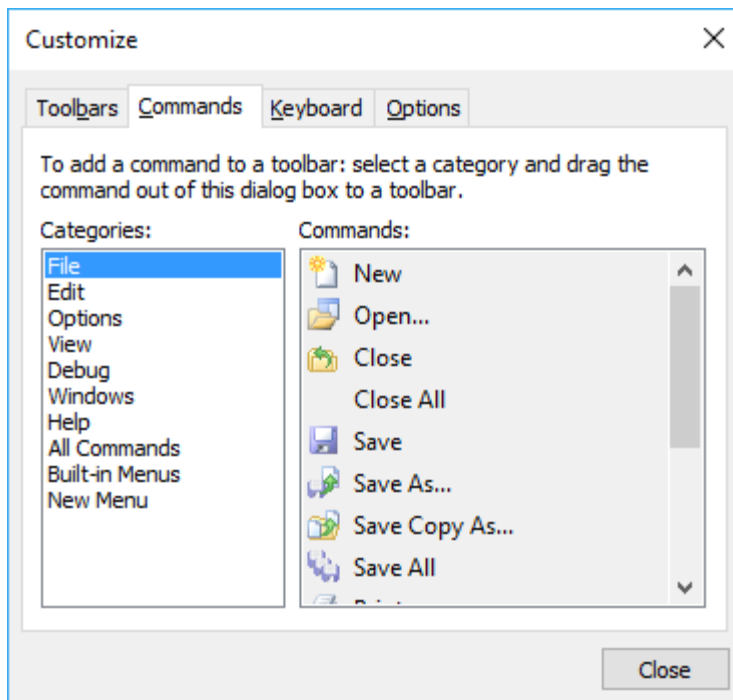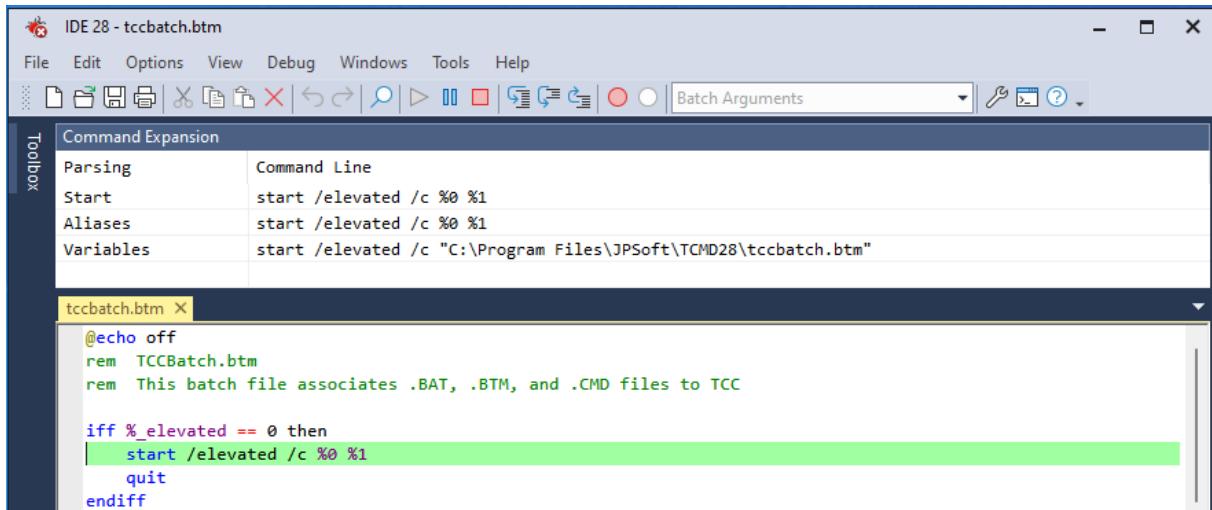

3.7   **Toolbox**

When you are using *TCC-RT* syntax, *CMDebug* will show a window that lists all of the internal commands, variables, and variable functions organized by category. Double-clicking on a command will insert the resulting command on the current line in the editor. Selecting a command or variable and then pressing F1 will display the help for that command / variable.

Toolbox

Commands and Variables
- Flow Control
  - Conditional command execution (IF)
  - If / Then / Else (IFF)
  - Switch / case (SWITCH)
  - FOR looping (FOR)
  - Do While / Until looping (DO)
  - Call batch subroutine (GOSUB)
  - Go to batch label (GOTO)
  - Call another batch file (CALL)
- Directories
- Files
- Keyboard Input
- Display Output
- Window Management
- Monitoring
- Batch files
- Environment and Path
- System Configuration and Status
- Internet
- LAN
- Compression / Decompression
- Tasks
- Video and Audio
- Miscellaneous
- Variables
- Functions

3.8   Command Expansion

The batch debugger has a "Command Expansion" window that will pop up above the tab window when you start debugging. The Command Expansion window will show the original command line, the command line after alias expansion, and the command line after variable expansion. The Command Expansion window is a docking window, so it can be moved & attached at other locations. If you don't want to see the Command Expansion window, you can turn it off from the IDE "View / Command Expansion" menu option.

3.9   Status Bar

The **CMDebug** window has a Status Bar that displays tooltips when you move the cursor over menu entries.
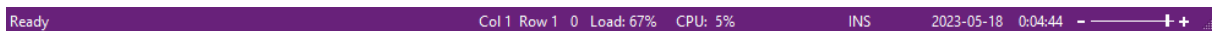
The status bar also displays the following information:

- The current cursor position (column and row)
- The Unicode value of the character at the current cursor location
- The edit 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 Insert key
- The state of the Scroll Lock key
- The current date
- The current time

You can hide the status bar fields by right clicking on the status bar and unchecking the fields you don't want to see.

There is a slider in the right corner that allows you to change the transparency level of the **CMDebug** window. You can also change the transparency with Ctrl-Shift-Mousewheel.

3.10   Edit Windows

The *CMDebug* edit windows allow you to edit and debug Windows batch files.

If a file in a tab edit window has been modified but not yet saved, the tab title will be prefixed with a *****. When the file is saved, the ***** is removed.

If you are using **TCC-RT syntax**, 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 also hover the mouse over a *TCC-RT* variable name, and *CMDebug* will pop up a tooltip with the current value. If you hover the mouse over a *TCC-RT* internal command name, *CMDebug* will pop up a tooltip with the command syntax.

You can change the line to be executed next when in debugging mode 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!

You can block select in the edit window by holding down the Alt key while selecting text with the left mouse button.

The edit window supports multiple selections at one time. You can select additional text by holding down the Ctrl key while dragging with the mouse. Multiple selections are added to the clipboard in order with no delimiting characters. For block selections, the line end is added after each line of text. Block selections are always copied from top to bottom, not in the order of selection.

The editor will display document changes in the margin and in the text. In the text, inserted characters appear with colored underlines and points where characters were deleted are shown with small triangles. The margin shows a block indicating the overall state of the line. The states are modified (orange), saved (green), saved then reverted to modified (green-yellow), and saved then reverted to original (cyan). The change history can be toggled on or off with the "Options / Change History" menu entry.

Margins

There are three possible margins on the left of the edit window:

- The line number (selectable by the "Options / Display Line Numbers" menu option).
- The Breakpoint margin (left click in this margin to set a breakpoint on this line).
- The Fold margin (selectable by the "Options / Display Fold Margin" menu option), which will display a - for blocks that can be collapsed to a single line (DO, IFF, and SWITCH commands, and command groups). When a block is collapsed, the Fold margin will display a **+**. Left clicking in the Fold margin will toggle the fold state.

Syntax Coloring

CMDebug will select the syntax lexer (colorization) based on the file extension:

```
.bat     CMD (or optionally TCC-RT)
.btm     TCC-RT
.cmd     CMD (or optionally TCC-RT)
.css     CSS
.htm     HTML
```

.html   HTML
.lua   Lua
.php   PHP
.pl   Perl
.ps1   PowerShell
.py   Python
.rb   Ruby
.sh   Bash shell
.sql   SQL
.tcl   Tcl/Tk
.vbs   VBScript
.xml   XML

The edit window toolbar (which is configurable by clicking on the rightmost down arrow), has a number of icons to control debugging. Each has a tooltip for quick reference:

New	Create a new batch file in a new tab window.
Open	Open an existing batch file in a new tab window.
Save	Save the current batch file.
Print	Print the current batch file.
Cut	Copy the highlighted selection to the clipboard and delete it from the file.
Copy	Copy the highlighted selection to the clipboard.
Paste	Copy the contents of the clipboard to the current cursor location.
Delete	Delete the highlighted selection.
Undo	Undo the last edit.
Redo	Restore the last Undo.
Find	Search for text.
Batch Arguments	New batch file arguments. The text will be parsed into %1 - %n batch arguments and used when the batch file is debugged.
Start Debugging	Starts the debugger. The cursor will be placed on the first line.
Pause Debugging	Pause execution at the next line.
Stop Debugging	Stops the debugger.
Step Into	Execute the current line.
Step Over	Execute the current line but disable the debugger during 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	Clear all breakpoints in the current batch file.
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.)
Help	Display the online help.

3.11   Editing Commands

Edit Windows

You can block select in the edit window by holding down the Alt key while selecting text with the left mouse button.

The edit window supports multiple selections. Select additional text by holding down the Ctrl key while dragging with the mouse. Multiple selections are added to the clipboard in order with no delimiting characters. For block selections, the line end is added after each line of text. Block selections are always copied from top to bottom, not in the order of selection.

The text processing commands available in the *CMDebug* edit windows are listed below. The text commands can be classified into general categories:

- ▸ [Caret commands](#)
- ▸ [Edit commands](#)
- ▸ [Mark / Clipboard commands](#)
- ▸ [Search commands](#)
- ▸ [File commands](#)
- ▸ [Bookmark commands](#)
- ▸ [Breakpoint commands](#)
- ▸ [Expression evaluation commands](#)

- **Caret commands**

Right	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	In addition to the caret movement this command will also extend the current selection to the new caret position.
Left	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	In addition to the caret movement, this will also extend the current selection to the new position.
Up	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	In addition to the caret movement this command will also extend the current selection to the new position.
Down	This command will move the caret one line down. The caret column position will be set as close to it's 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	In addition to the caret movement this command will also extend the current selection to the new position.
End	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	In addition to the caret movement this command will also extend the current selection to the new position.
Home	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	In addition to the caret movement this command will also extend the current selection to the new position.
Ctrl-Right	This command will move in one of the following ways: • When the caret is located on a delimiter character the caret is moved right until the first non-delimiter is found. • When the caret is located on a non-delimiter character the caret is moved to the next delimiter character. • When the caret is located on the last word or delimiter of the current line the caret is moved to the first word or delimiter of the next line.
Ctrl-Shift-Right	In addition to the caret movement this command will also extend the current selection to the new caret position.
Ctrl-Left	This command will move in one of the following ways:

- When the caret is located on a delimiter character the caret is moved to the start of the previous word.
- 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 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	In addition to the caret movement this command will also extend the current selection to the new position.
Ctrl-Home	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	In addition to the caret movement this command will also extend the current selection to the new position.
Ctrl-End	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	In addition to the caret movement this command will also extend the current selection to the new position.
PgUp	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	In addition to the caret movement this command will also extend the current selection to the new position.
PgDn	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	In addition to the caret movement, this command will also extend the current selection to the new position.

- **Edit commands**

Ctrl-Z	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	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.
Backspace	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	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.
Return	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.
Ctrl-Delete	When the caret is located on a word, this command will delete all characters in the word right of the caret position.
Ctrl-Backspace	When the caret is located on a word, this command will delete all characters in the word left of the caret position.
Tab	This command does one of the two following things:

- 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	When there is a valid text selection, this command will indent the lines covered by the selection left by one tab-stop.
Shift-Ctrl-U	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	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.
Ins	This command will toggle the current editing mode between overwrite and insert.

- **Mark / Clipboard commands**

Ctrl-A	This command will select all the text.
Ctrl-V	This command will, when there is text present in the clipboard, paste the clipboard contents at the current position.
Ctrl-C	This command will, when there is a selection, copy the selected text to the clipboard.
Ctrl-X	This command will, when there is a selection, copy the selected text to the clipboard and remove the selection from the text.

- **Search commands**

Ctrl-F3	This command will find the next occurrence of the word under the caret. When the next occurrence is found, it is selected.
F3	This command will find the next occurrence of the current search pattern. When the search pattern is found, it is selected.
Shift-F3	This command will find the previous occurrence of the current search pattern. When the search pattern is found, it is selected.
Ctrl-G	This command will show the *goto* dialog.
Ctrl-F	This command will show the *find* dialog.
Ctrl-H	This command will show the *replace* dialog.

- **File commands**

Ctrl-N	Open a new file in a new tab window.
Ctrl-O	Open an existing file in a new tab window.
Ctrl-W	Close all files.
Ctrl-S	This command will save the current file.
Ctrl-Shift-S	Save all files.
Ctrl-P	This command will open the print dialog.
Ctrl-I	Display the properties for the current file.
Alt-F4	Exit the debugger.

- **Bookmark commands**

Ctrl-F2	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	This command will clear all bookmarks.
F2	This command will place the caret on the next line which has a bookmark set. When there is no next line with a bookmark, the text is searched starting at the first line.
Shift-F2	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-F9   This command will toggle a breakpoint on the current line.
  Ctrl-Shift-F9   This command will clear all breakpoints.

- **Expression evaluation commands**

  Alt-F11   Invoke the Evaluate Expression dialog.
  Alt-Shift-F11   Invoke the Evaluate Expression dialog for the current selection.

                         You can select the result and copy it to the clipboard.

3.12   Watch Tab Window

The Watch window allows you to monitor environment variables, internal variables, variable functions, and user-defined functions, 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 %.

If you right click on the first column in the Watch window, the debugger will display an environment variable listbox. Select an entry to have it added to the watch list.

When the value of a monitored variable changes, the Watch window will change the text color to red.

The Watch tab default to always showing two variables:

  %_?   The last *TCC-RT* internal command result
  %?   The last ERRORLEVEL value

The watch windows has a toolbar, with the following buttons:

  New   Restore the original values for the watch list
  Open   Add the contents of a file to the watch list
  Save   Save the current watch list to a file
  Apply   Replace the original values with the modified watch list
  Print   Print the current watch list
  Cut   Copy the highlighted selection to the clipboard and delete it from the watch list
  Copy   Copy the highlighted selection to the clipboard
  Paste   Copy the contents of the clipboard to the current cursor location
  Delete   Delete the highlighted selection (or the character at the cursor location if no selection)

Undo	Undo the last edit
Redo	Restore the last Undo
Find	Search for text
Help	Display the online help

The debugger will automatically save watch lists (the current batch file name + ".watch"), and reload them the next time the batch file is loaded in the debugger.

3.13   Modified Tab Window

The Modified variables window shows all variables that are created or modified while executing the batch file. (This is like the "Auto" window in Visual Studio.) The window will show the variable name, the previous value, and the new value.

The Modified window has a toolbar, with the following buttons:

New	Restore the original values for the modified variables list
Save	Save the current modified variables list to a file
Apply	Replace the original values with the modified list
Print	Print the current modified variables list
Cut	Copy the highlighted selection to the clipboard and delete it from the modified variables list
Copy	Copy the highlighted selection to the clipboard
Paste	Copy the contents of the clipboard to the current cursor location
Delete	Delete the highlighted selection (or the character at the cursor location if no selection)
Undo	Undo the last edit
Redo	Restore the last Undo
Find	Search for text
Help	Display the online help

3.14   Breakpoints Tab Window

CMDebug debugging will pause when it reaches a line with a breakpoint set.

You can set a breakpoint either through the menu, the toolbar, or 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 define conditional breakpoints by specifying the number of iterations before the breakpoint is triggered, and/or define a conditional expression that must be true before the breakpoint is triggered. After setting the breakpoint, enter the conditions either by right-clicking on the breakpoint and selecting "Break >=" (to set the minimum number of iterations), or "Condition" (to set the conditional expression). You can also select the Breakpoint window and double-click on the "Break >=" or "Condition" columns to edit or modify the conditions.

You can temporarily disable a breakpoint (without deleting it) by right-clicking on the breakpoint and selecting "Enable/Disable Breakpoint". You can disable all breakpoints by clicking on the Debug menu and selecting "Disable All Breakpoints".

The breakpoint window has a toolbar, with the following buttons:

New	Restore the original values for the breakpoints list
Open	Add the contents of a file to the breakpoints list
Save	Save the current breakpoints list to a file

Apply	Replace the original values with the modified breakpoints list
Print	Print the current breakpoints list
Cut	Copy the highlighted selection to the clipboard and delete it from the breakpoints
Copy	Copy the highlighted selection to the clipboard
Paste	Copy the contents of the clipboard to the current cursor location
Delete	Delete the highlighted selection (or the character at the cursor location if no selection)
Undo	Undo the last edit
Redo	Restore the last Undo
Find	Search for text
Help	Display the online help

The debugger will automatically save breakpoints (the current batch file name + ".bp"), and reload them the next time the batch file is loaded in the debugger.

3.15   Environment Tab Window

The environment window displays the Windows environment used when debugging the batch file in the active edit window.

The environment window has a toolbar, with the following buttons:

New	Restore the original values for the environment
Open	Add the contents of a file to the environment
Save	Save the current environment to a file
Apply	Replace the original values with the modified environment
Print	Print the current environment
Cut	Copy the highlighted selection to the clipboard and delete it from the environment
Copy	Copy the highlighted selection to the clipboard
Paste	Copy the contents of the clipboard to the current cursor location
Delete	Delete the highlighted selection (or the character at the cursor location if no selection)
Undo	Undo the last edit
Redo	Restore the last Undo
Find	Search for text
Help	Display the online help

3.16   Batch Parameters Tab Window

The Batch Parameters window shows the batch file arguments (%0 - %9). You can enter default arguments using the batch arguments combo box on the toolbar. (You can change them before running the batch file.)

The Batch Variables window supports modifying any of the batch arguments during batch file execution.

The batch window has a toolbar, with the following buttons:

New	Restore the original values for the batch parameters
Open	Add the contents of a file to the batch parameters list
Save	Save the current batch parameters list to a file
Apply	Replace the original values with the modified batch parameters list
Print	Print the current list

Cut	Copy the highlighted selection to the clipboard and delete it from the batch parameters
Copy	Copy the highlighted selection to the clipboard
Paste	Copy the contents of the clipboard to the current cursor location
Delete	Delete the highlighted selection (or the character at the cursor location if no selection)
Undo	Undo the last edit
Redo	Restore the last Undo
Find	Search for text
Help	Display the online help

3.17   Aliases Tab Window

(TCC-RT Syntax Only)

The Local and Global Aliases tab windows allows you to display and/or modify *TCC-RT* user-defined functions.

CMDebug will only display the Local Alias tab window if you have local aliases defined, and the Global Alias tab window if you have global aliases defined.

The aliases windows have a toolbar, with the following buttons:

New	Restore the original values for the alias list
Open	Add the contents of a file to the alias list
Save	Save the current alias list to a file
Apply	Replace the original values with the modified alias list
Print	Print the current alias list
Cut	Copy the highlighted selection to the clipboard and delete it from the alias list
Copy	Copy the highlighted selection to the clipboard
Paste	Copy the contents of the clipboard to the current cursor location
Delete	Delete the highlighted selection (or the character at the cursor location if no selection)
Undo	Undo the last edit
Redo	Restore the last Undo
Find	Search for text
Help	Display the online help

3.18   Functions Tab Window

(TCC-RT Syntax Only)

The Local and Global Functions tab windows allows you to display and/or modify *TCC-RT* user-defined functions.

TCC-RT will only display the Local Functions tab window if you have local functions defined, and the Global Functions tab window if you have global functions defined.

The Functions windows have a toolbar, with the following buttons:

New	Restore the original values for the user-defined functions list
Open	Add the contents of a file to the user-defined functions list
Save	Save the current user-defined functions list to a file
Apply	Replace the original values with the modified user-defined functions list
Print	Print the current user-defined functions list

Cut	Copy the highlighted selection to the clipboard and delete it from the user-defined functions list
Copy	Copy the highlighted selection to the clipboard
Paste	Copy the contents of the clipboard to the current cursor location
Delete	Delete the highlighted selection (or the character at the cursor location if no selection)
Undo	Undo the last edit
Redo	Restore the last Undo
Find	Search for text
Help	Display the online help

3.19   CMD.EXE Comparison

The comparison of commands available is based on the version of **CMD** included with Windows 10.

If the CMD command name matches an internal *TCC-RT* command, the *TCC-RT* command is almost always substantially enhanced.

▸   TCC-RT, TCC/LE, and CMD commands
▸   Redirection
▸   Wildcards
▸   Built-In Variables
▸   Unique TCC features

TCC, TCC-RT, and CMD commands

TCC	*TCC-RT*	*CMD*
?	Y	
ACTIVATE	Y	
ALIAS	Y	
ASSOC	Y	Y
ASSOCIATE	Y	
ATTRIB	Y	*
BATCOMP		
BDEBUGGER		
BEEP	Y	
BTMONITOR	Y	
BREAK	Y	Y
BREAKPOINT	Y	
BZIP2	Y	
CALL	Y	Y
CANCEL	Y	
CD / CHDIR	Y	Y
CDD	Y	
CHCP	Y	*
CHRONIC	Y	
CLIPMONITOR	Y	
CLS	Y	Y
COLOR	Y	Y
COPY	Y	Y
COPYDIR	Y	
DATE	Y	Y

DATEMONITOR	Y	
DEBUGMONITOR	Y	
DEBUGSTRING	Y	
DEDUPE	Y	
DEFER	Y	
DEL / ERASE	Y	Y
DELAY	Y	
DESCRIBE	Y	
DESKTOP	Y	
DETACH	Y	
DIFFER	Y	
DIR	Y	Y
DIRENV	Y	
DIRHISTORY		
DIRS	Y	
DISKMONITOR	Y	
DO	Y	
DRAWBOX	Y	
DRAWHLINE	Y	
DRAWVLINE	Y	
ECHO	Y	Y
ECHOERR	Y	
ECHOS	Y	
ECHOSERR	Y	
ECHOX	Y	
ECHOXERR	Y	
EJECTMEDIA	Y	
ENDLOCAL	Y	Y
ENUMPROCESSES	Y	
ENUMSERVERS	Y	
ENUMSHARES	Y	
ESET	Y	
EVENTLOG	Y	
EVENTMONITOR	Y	
EVERYTHING	Y	
EXCEPT	Y	
EXIT	Y	Y
EXPR	Y	
FFIND	Y	
FIREWIREMONITOR	Y	
FOLDERMONITOR	Y	
FONT	Y	
FOR	Y	Y
FREE	Y	
FTYPE	Y	Y
FUNCTION	Y	
GLOBAL	Y	
GOSUB	Y	
GOTO	Y	Y
GZIP	Y	
HASH	Y	

HEAD	Y	
HELP		*
HISTORY		
IDE		
IF	Y	Y
IFF	Y	
IFTP	Y	
INKEY	Y	
INPUT	Y	
INSTALLED	Y	
JABBER	Y	
JAR	Y	
JOBMONITOR	Y	
JOBS	Y	
JUMPLIST		
KEYBD	Y	
KEYS	Y	Y
KEYSTACK	Y	
LIST	Y	
LOADBTM	Y	
LOADMEDIA	Y	
LOCAL	Y	
LOCKMONITOR	Y	
LOG	Y	
LUA	Y	
MD / MKDIR	Y	Y
MEMORY	Y	
MKLINK	Y	Y
MKLNK	Y	
MONITOR	Y	
MOUNTISO	Y	
MOUNTVHD	Y	
MOVE	Y	Y
MOVEDIR	Y	
MSGBOX	Y	
NETMONITOR	Y	
ON	Y	
OPTION	Y	
OSD	Y	
PATH	Y	Y
PAUSE	Y	Y
PDIR	Y	
PEE	Y	
PIPEVIEW	N	
PLAYAVI	Y	
PLAYSOUND	Y	
PLUGIN	Y	
POPD	Y	Y
POSTMSG	Y	
PRINT	Y	
PRIORITY	Y	

PROCESSMONITOR	Y	
PROMPT	N	Y
PSHELL	Y	
PSUBST	Y	
PUSHD	Y	Y
QUERYBOX	Y	
QUIT	Y	
RD / RMDIR	Y	Y
REBOOT	Y	
RECORDER		
RECYCLE	Y	
REGDIR	Y	
REGMONITOR	Y	
REM	Y	Y
REN / RENAME	Y	Y
RESOLUTION	Y	
RESTOREPOINT	Y	
RETURN	Y	
REXEC	Y	
RSHELL	Y	
SAVECONSOLE	Y	
SCREEN	Y	
SCREENMONITOR	Y	
SCRIPT	Y	
SCRPUT	Y	
SELECT	Y	
SENDHTML	Y	
SENDMAIL	Y	
SERVICEMONITOR	Y	
SERVICES	Y	
SET	Y	Y
SETARRAY	Y	
SETDOS	Y	
SETERROR	Y	
SETLOCAL	Y	Y
SHIFT	Y	Y
SHORTCUT	Y	
SHRALIAS	Y	
SMPP	Y	
SNMP	Y	
SNPP	Y	
SPONGE	Y	
SREPLACE		
SSHEXEC		
START	Y	Y
STATUSBAR		
SWITCH	Y	
SYNC	Y	
TABCOMPLETE		
TAIL	Y	
TAR	Y	

TASKBAR	Y	
TASKDIALOG	Y	
TASKEND	Y	
TASKLIST	Y	
TCDIALOG		
TCFILTER		
TCTOOLBAR		
TEE	Y	
TEXT	Y	
TIME	Y	Y
TIMER	Y	
TITLE	Y	Y
TOUCH	Y	
TPIPE	Y	
TRANSIENT	Y	
TREE	Y	*
TRUENAME	Y	
TS	Y	
TYPE	Y	Y
UNALIAS	Y	
UNBZIP2	Y	
UNFUNCTION	Y	
UNGZIP	Y	
UNJAR	Y	
UNMOUNTISO	Y	
UNMOUNTVHD	Y	
UNQLITE	Y	
UNSET	Y	
UNSETARRAY	Y	
UNTAR	Y	
UNZIP	Y	
UPTIME	Y	
USBMONITOR	Y	
UUID	Y	
VBEEP	Y	
VDESKTOP	Y	
VER	Y	Y
VERIFY	Y	Y
VIEW		
VOL	Y	Y
VSCRPUT	Y	
WAITFOR	Y	
WAKEONLAN	Y	
WEBFORM	Y	
WEBSOCKET	Y	
WEBUPLOAD	Y	
WHICH	Y	
WINDOW	Y	
WINSTATION	Y	
WMIQUERY	Y	
WMIRUN	Y	

WSETTINGS	Y	
WSHELL	Y	
WSHORTCUT	Y	
Y	Y	
ZIP	Y	
ZIPSFX	Y	
7UNZIP	Y	
7ZIP	Y	

* This is an internal command in **TCC-RT** but an external command in CMD.

Redirection

In addition to the CMD <, > and |, **TCC-RT** allows you to also redirect standard error, combine standard output and standard error, protect existing files from being overwritten by redirection, and redirect standard input using "here-documents". See Redirection for more details.

Wildcards

CMD only supports the ? and * wildcards in filenames. **TCC-RT** adds character sets and regular expressions, and also supports wildcards in pathnames. See Wildcards for more details.

Built-In Variables

CMD has a few 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-RT supports all of these built-in variables. (In **TCC-RT**, **CMDEXTVERSION** will always return **2**.) **TCC-RT** also includes 220+ additional internal variables, 380+ variable functions, and 60+ command variables.

Unique **TCC-RT** features

TCC-RT includes many more features not in CMD, including:

Aliases

Internal functions

User defined functions

File selection

File Ranges

Conditional Commands

Internet access and email

OpenAFS support

Logs

Intersession sharing

Lua, **Perl**, **Python**, **REXX**, and **Tcl** support

3.20   CMD Compatibility

We try to keep **CMDebug and** TCC-RT as compatible as possible with CMD, given the limitations and bugs in CMD, the variances in CMD in different versions of Windows, and the thousands of additional features available in **TCC-RT**.

In the **CMDebug** Options menu, there is an option to choose either **TCC** syntax or CMD syntax. (The default for .BAT and .BTM files is CMD syntax.) If you choose CMD syntax, **CMDebug** will disable the **TCC**-only commands, internal variables, variable and user-defined functions, and plugins. **CMDebug** will also set a number of options to maximize CMD compatibility, including emulating a number of (known) CMD bugs.

On rare occasions, you may find batch files that exploit undocumented features (or bugs) in CMD (or are simply badly written) that don't work in **CMDebug**. If you do, please send those batch files to *support@jpsoft.com* so we can try to emulate the CMD behavior (or at least suggest a better workaround).

3.21   Uninstalling CMDebug

Before uninstalling **CMDebug**, if you have registered it then click on the **Help** menu, and select **Register**. Enter your original activation key, and click on the **Unregister** button*,* and your registration info will be removed from the activation server, and from the local machine. This will prevent the local machine from continuing to use one of your activations. (A single-system license is allowed to activate up to three systems.)

You can uninstall **CMDebug** from the "Programs and Features" option in the Windows Control Panel.

The easiest way to remove these files is from a **TCC** command line before uninstalling **CMDebug**.

You can remove TCMD.INI with the command:

```
del %_ininame
```

You can remove TCSTART with the command:

```
del %_tcstart
```

You can remove TCEXIT with the command:

```
del %_tcexit
```

CMDebug also writes some user-defined configuration options to the "c:\ProgramData\JP Software" directory. You can remove everything in that directory.

After uninstalling, *CMDebug* will show an optional uninstall survey. Please take a moment to respond -- Your answers will help improve *CMDebug*.

3.22   Updater

You can check for updates to *CMDebug* with the Help / Check for Updates menu entry.

You can also automate updates by using the Updater tool. UPDATER.EXE is a small executable tool (located in your *CMDebug* installation directory) whose role is to check for updates, inform the user of their presence and offer to download and install them. When launched, the Updater checks if a newer version of *CMDebug* exists.

Updater options

You can also run the Updater from the command line or batch files. The Updater has the following command line options:

- /checknow - The Updater is launched, pops up a dialog box, checks for updates and automatically informs the user that new updates are available. If no updates are available, the Updater will exit immediately.
- /silent - The Updater will search silently for updates at the interval specified by the user. The search interval can be specified in the Configuration dialog. By default it is the value you specified in the "Check Frequency" field (Updater Page). If the check frequency has not passed or there are no updates available, the Updater will exit immediately.
- /silentall - The Updater will search silently for updates and automatically install all updates. This has the has the same effect as the /silent option if the user has selected the "Check and automatically install all updates" option in the configuration dialog. If the check frequency has not passed or there are no updates available, the Updater will exit immediately.
- /configure - the Configure dialog will be displayed allowing the configuration of the Updater. In this dialog the user can specify the download folder, the check frequency, enable or disable the automatically update option.

4   TCC-RT

TCC-RT is a command processor compatible with CMD (the default command processor in Windows) but massively enhanced with thousands of additional features. *TCC-RT* is a runtime-only version of *TCC*, and is distributed as a separate free product.

- Starting TCC
- Commands
- Variables & Functions
- Aliases & Batch Files
- File Selection
- Input / Output Redirection

4.1   Starting TCC-RT

If you are using **TCC-RT Syntax,** *CMDebug* will automatically start *TCC-RT* in a console window. But you can also start *TCC-RT* 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.

See TCC Startup Options for more information on startup command line options.

When you configure a TCC-RT shortcut, place the full path and name for the file in the Command Line field, and put any startup options that you want passed to *TCC-RT.* For example:

Command Line: `C:\Program Files\JPSoft\TCC_RT_31\TCC.EXE`
Working directory: `C:\`

You do not need to use the Change Icon button, because *TCC.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.

4.1.1   TCC Startup Options

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

```
"c:\program files\jpsoft\tcmd17\tcc.exe" @c:\jpsoft\tcmd.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 command line prompt or in a batch file (with or without using the internal START 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-RT* is started from *TCC-RT* without the START command.

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

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

```
d:\path\tcc.exe [d:\path][[/]@d:\path\inifile][//directive=value...]
   [/A /H /I[IPSX]/L: /LA /LD /LF /LH /N/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-RT** 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-RT** 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. You don't need this option if

1)   your .INI file is named *TCMD.INI,* and
2)   it is in one of the following directories:
    2.1)   the same directory as **TCC-RT**
    2.2) the "%programdata%\JP Software\CMDebug 21" directory
    2.3)   the %localappdata% directory

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

To start **TCC-RT** without any .INI file, you can use the /I or /II options, or create an empty file and specify it as your .INI file.

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-RT will skip the leading slash.

```
//directive=value
```

This option tells **TCC-RT** 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, 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.

Directives on the command line override any corresponding directive in the .INI file.

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

/B   This option tells **TCC-RT** that you do not want it to set up a Ctrl-C / Ctrl-Break handler.

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.

/D   Disable execution of AutoRun commands from Registry. If /D is not specified when **TCC-RT** 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

/H   Start **TCC-RT** 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 TCEXIT, or load plugins. You can optionally specify individual arguments:

        /II   Don't load the .INI file.
        /IP   Don't load plugins.
        /IS   Don't execute TCSTART.
        /IX   Don't execute TCEXIT.

/L: Forces the use of local lists as the default 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 and FUNCTION for more details. Note the required trailing colon (:)!

You can optionally specify individual arguments:

        /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.

/N   If **TCC** was started as a service, use the /N option to prevent **TCC** from being closed on a Windows CTRL_LOGOFF_EVENT.

/Q   Don't display version / copyright message (registered copies only).

/S   CMD.EXE-compatible quoted string handling.

/T:*bf*   This option sets the foreground and background colors in the **TCC-RT** 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*.* See the CMD color codes in Colors, Color Names & Codes.

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

OPTION //UnicodeOutput=yes | no

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

/U8   This option causes the output of internal commands to a pipe or redirected to a file to be in UTF-8 when *TCC-RT* starts. The command :

OPTION //UTF8Output=yes | no

may be used at any time to switch between UTF-8 and ASCII output.

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

/X   This option forces *TCC-RT* to alter the operation of the MD (MKDIR) command to automatically create all necessary intermediate directories when it creates a new subdirectory. Its effect is the same as adding a /S option to all MD (MKDIR) commands. This option is included for compatibility with CMD, where it also enables other options. However, in *TCC-RT* 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-RT* 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, 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-RT* will treat characters after *command* as parameters for *command* and not as additional startup options.

If *command* is preceded by */C*, *TCC-RT* 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.

Example 1

Assume that you execute the command line below:

```
c:\TCMD\TCC.exe c:\TCMD\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 "%programdata\JP Software\CMDebug 21" directory
       3rd 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-RT.

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-RT*

4 *TCC* executes *c:\tcmd\start.btm* (or, if not found, it displays an error message).

5 *TCC* displays the command prompt, unless an EXIT command was executed in *c:\tcmd\start.btm*, terminating TCC-RT.

Example 2

The command line below, when executed by *TCC-RT*, CMD, the RUN dialog, or a shortcut, will start *TCC-RT*, 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:\tcmd13\tcc.exe /la /c c:\tcmd13\process.btm
```

4.1.2   TCC Exit Codes

If you start TCC-RT 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-RT's exit code is normally the numeric exit code from the last internal or external command. However, for CMD 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 *n* command to explicitly set the exit code. This overrides the rules above, and sets the return code to the parameter of your EXIT command.

4.1.3   TCSTART and TCEXIT

TCC-RT Startup Program

Each time TCC-RT starts, it looks for a program named TCSTART. TCSTART is normally a batch file (*.BAT*, *.BTM*, or *.CMD*), but it can be any executable file. 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-RT will not display an error message if it cannot find the program.

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

With the exception of some initialization switches, the entire startup command line passed to TCC-RT is available to *TCSTART* as batch file parameters (**%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 [%$]
```

You can disable TCSTART and/or TCEXIT

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 or when a transient copy of TCC-RT is started with the **/C** startup option. 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-RT 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\tcc.exe /c list myfile.txt`
Working Directory: `c:\data`

Normally this shortcut would LIST the file *C:\DATA\MYFILE.TXT*. However, if *TCSTART* changes the default to a different directory, TCC-RT 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-RT, including those used for pipes and transient sessions.

You can work around these potential problems with the IF or IFF commands and the _PIPE and _TRANSIENT 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-RT Termination Program

Whenever a TCC-RT session ends, it looks for a program named TCEXIT. *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-RT exits, or to LOG 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.

4.2   Commands

TCC-RT gives you instant access to more than 240 internal commands. (By contrast, Microsoft's CMD 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

▸   Commands By Category

Note: Remember that you can replace any internal command with an ALIAS or plugin, or disable an internal command with SETDOS /I.

4.2.1   Commands by Name

See also: Internal Commands Listed by Category

	Description
?	Display list of internal commands or Prompt to execute a command
ACTIVATE	Activate or set window state
ALIAS	Define or display aliases
ASSOC	Windows file associations
ASSOCIATE	Combine ASSOC and FTYPE
ATTRIB	Change or display file attributes
BEEP	Beep the speaker
BREAK	Define or display Ctrl-C state
BREAKPOINT	Set a batch debugger breakpoint
BTMONITOR	Monitor Bluetooth connections
BZIP2	Create bz2 archives
CALL	Call another batch file
CANCEL	End batch file processing
CAPTURE	Video and/or audio capture
CD	Display or change directory
CDD	Change drive and directory
CHCP	Display or change code page
CHDIR	Display or change directory
CHRONIC	Run command and hide STDOUT & STDERR
CLIP	Display or modify *TCC-RT* clipboards
CLIPMONITOR	Monitor Windows clipboard
CLS	Clear the display window
COLOR	Change the display colors
COMMENT	Enter multiline comments
COPY	Copy files and/or directories
COPYDIR	Copy directory tree
DATE	Display or change date
DATEMONITOR	Monitor the current date and time
DEBUGMONITOR	Monitor OutputDebugString calls
DEBUGSTRING	Send text to system debugger
DEDUPE	Search for duplicate files
DEFER	Defer a command until batch file exit
DEL	Delete files and/or directories
DELAY	Wait for specified time
DESCRIBE	Display or change descriptions
DESKTOP	Create or switch desktops

DETACH	Start app detached
DIFFER	Show directory differences
DIR	Display files and/or directories
DIRENV	Configure environment per directory
DIRS	Display directory stack
DISKMONITOR	Monitor disk usage
DO	Create batch file loops
DRAWBOX	Draw a box
DRAWHLINE	Draw a horizontal line
DRAWVLINE	Draw a vertical line
ECHO	Echo a message
ECHOERR	Echo a message to STDERR
ECHOS	Echo a message with no CR/LF
ECHOSERR	Echo with no CR/LF to STDERR
ECHOX	Echo with no expansion to STDOUT
ECHOXERR	Echo with no expansion to STDERR
EJECTMEDIA	Eject a removable drive
ENDLOCAL	Restore from a SETLOCAL
ENUMPROCESSES	Enumerate child processes
ENUMSERVERS	Enumerate network servers
ENUMSHARES	Enumerate network sharenames
ERASE	Delete files and/or directories
ESET	Edit variables or aliases
EVENTLOG	Write Windows event log
EVENTMONITOR	Monitor event log
EXCEPT	Exclude files from a command
EXIT	Exit *TCC*
EXPR	Evaluate expressions
FFIND	Search for files or text
FILELOCK	Show file locks
FIREWIREMONITOR	Monitor FireWire devices
FOLDERMONITOR	Monitor folders and/or files
FONT	Change console font
FOR	Repeat a command
FREE	Display disk space
FTYPE	Display or edit file types
FUNCTION	Create or edit user functions
GLOBAL	Run command in subdirectories
GOSUB	Call batch subroutines
GOTO	Branch in a batch file
GZIP	Compress files to .gz archive
HASH	Display file hash values
HEAD	Display beginning of file
IF	Conditional command execution
IFF	Conditional command execution
IFTP	Open FTP connection

INKEY	Get a single keystroke
INPUT	Get a text string
INSTALLED	Show installed applications
JABBER	Send an IM
JAR	Create Java JAR archive
KEYBD	Set keyboard toggles
KEYS	Enable or disable history list
KEYSTACK	Send keystrokes to app
LIBRARY	Load, display, or delete library functions
LIST	Display content of files
LOADBTM	Load batch file as .BTM
LOADMEDIA	Close CD-ROM / DVD drive door
LOCAL	Local variables for batch files & library functions
LOCKMONITOR	Monitor session locking / unlocking
LOG	Save log of commands
LUA	Call the internal Lua interpreter
MD	Create subdirectories
MEMORY	Display memory statistics
MKDIR	Create subdirectories
MKLINK	Create NTFS symbolic links
MKLNK	Create NTFS hard or soft link
MOUNTISO	Mount ISO disk
MOVE	Move files or directories
MOVEDIR	Move directory tree
MSGBOX	Popup message box
NETMONITOR	Monitor networks
ODBC	SQL database query
ON	Batch file error trapping
OPTION	Configure the *TCC* console
OSD	Display floating text
PATH	Set or display PATH
PAUSE	Wait for input
PDIR	User-formatted DIR
PEE	Redirect STDOUT to multiple pipes
PLAYAVI	Display an .AVI file
PLAYSOUND	Play a sound file
PLUGIN	Load or unload plugin DLL
POPD	Restore from directory stack
POSTMSG	Send a message to a Window
POWERMONITOR	Monitor system power
PRINT	Print a file
PRINTF	Formatted output
PRIORITY	Set process priority
PROCESSMONITOR	Monitor processes
PSUBST	Persistent SUBST
PUSHD	Save directory to stack

QUERYBOX	Popup input box
QUIT	Exit batch file
RD	Remove subdirectory
REBOOT	Reboot system
RECYCLE	Display or empty recycle bin
REGMONITOR	Monitor Windows Registry keys
RESOLUTION	Change display resolution
RESTOREPOINT	Create / delete / list system restore points
REXEC	Remotely execute commands
REM	Remark
REN	Rename files or directories
RENAME	Rename files or directories
REPEAT	Execute a counted loop
RETURN	Return from GOSUB
RMDIR	Remove subdirectory
RSHELL	Remotely execute commands
SCREEN	Position cursor
SCREENMONITOR	Monitor screen saver
SCRPUT	Write directly to screen
SELECT	Select files for a command
SENDHTML	Send HTML email
SENDMAIL	Send email
SERVICEMONITOR	Monitor Windows services
SERVICES	Display, stop, or start system services
SET	Set or display environment variables
SETDOS	Set or display console options
SETLOCAL	Save environment, aliases & functions
SETP	Set environment variable in another process
SHIFT	Shift batch file parameters
SHORTCUT	Create a Windows shortcut
SHRALIAS	Share aliases
SMPP	Simple message transfer
SNMP	Send SNMP traps
SNPP	Send message to pager
SPONGE	Read STDIN and write to file
SSHEXEC	SSH to remote host & run shell
START	Start a new session
SWITCH	Batch file switch / case
SYNC	Synchronize directories
TAIL	Display end of file
TAR	Add files to .tar archive
TASKBAR	Call Windows Taskbar functions
TASKDIALOG	Popup Windows task dialog
TASKEND	End a task
TASKLIST	Display Windows task list
TEE	Pipe "tee-fitting"

TEXT	Display text in batch file
TIME	Set or display time
TIMER	Stopwatch
TITLE	Set window title
TOUCH	Change file timestamps
TPIPE	Text filtering and substitution
TRANSIENT	Toggle shell transient mode
TREE	Display directory tree
TRUENAME	Display true pathname
TS	Timestamp pipe output
TYPE	Display files
UNALIAS	Remove aliases
UNBZIP2	Uncompress bz2 archives
UNFUNCTION	Remove user-defined functions
UNSET	Remove environment variable
UNSETP	Remove environment variable in another process
UNGZIP	Extract files from .gz archive
UNJAR	Extract files in a Java JAR archive
UNTAR	Extract files from .tar archive
UNMOUNTISO	Unmount ISO
UNZIP	Unzip files from zip archive
USBMONITOR	Monitor USB devices
VBEEP	Flash the screen and beep
VDESKTOP	Manage Windows 10 virtual desktops
VER	Display version
VERIFY	Display or set disk verification
VOL	Display or set disk volume label
VSCRPUT	Write text vertically
WAITFOR	Wait for app exit or input idle
WAKEONLAN	Send "Wake-On-LAN" packet
WEBFORM	Post data to web forms
WEBSOCKET	Connect to Websocket and send string
WEBUPLOAD	Upload files to web servers
WHICH	Display command information
WINDOW	Window management
WMIQUERY	WMI queries
WMIRUN	Run WMI methods
Y	Pipe "y-fitting"
ZIP	Zip files to zip archive
ZIPSFX	Create self-extracting executable
7UNZIP	Extract files from 7Zip archive
7ZIP	Compress files to 7Zip archive

4.2.2   Commands by Category

See also:

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](#)
- ▶ [Subdirectory management](#)
- ▶ [Input and output](#)
- ▶ [Window management commands](#)
- ▶ [Commands primarily for use in or with batch files and aliases](#)
- ▶ [Environment and path commands](#)
- ▶ [System configuration and status](#)
- ▶ [Monitoring commands](#)
- ▶ [Compression / Decompression](#)
- ▶ [Other commands](#)

File and directory management

	Description
ATTRIB	Change or display file attributes
COPY	Copy files and/or directories
COPYDIR	Copy directory tree
DEDUPE	Delete / Link duplicate files
DEL	Delete files and/or directories
DESCRIBE	Display or change descriptions
DIFFER	Show differences between directories
ERASE	Delete files and/or directories
FFIND	Search for files or text
FILELOCK	Show / release file locks
HEAD	Display beginning of file
IFTP	Open FTP connection
LIST	Display contents of files
MOVE	Move files or directories
MOVEDIR	Move directory tree
PSUBST	Persistent SUBST
RECYCLE	Display or empty recycle bin
REN	Rename files or directories
RENAME	Rename files or directories
SELECT	Select files for a command
SYNC	Synchronize directories
TAIL	Display end of file
TOUCH	Change file dates/times
TPIPE	Text filtering and substitution
TREE	Display directory tree
TRUENAME	Display true pathname
TYPE	Display files
UNZIP	Unzip files from archive
Y	Pipe "y-fitting"
ZIP	Zip files to archive

Subdirectory management

	Description
CD	Display or change directory
CDD	Change drive and directory
CHDIR	Display or change directory
DIR	Display files and/or directories
DIRS	Display directory stack
MD	Create subdirectories
MKDIR	Create subdirectories
MKLNK	Create NTFS hard or soft link
PSUBST	Persistent SUBST
PDIR	User-formatted DIR
POPD	Restore from directory stack
PUSHD	Save directory to stack
RD	Remove subdirectory
RMDIR	Remove subdirectory

Input and output

	Description
CAPTURE	Video and/or audio capture
DRAWBOX	Draw a box
DRAWHLINE	Draw a horizontal line
DRAWVLINE	Draw a vertical line
ECHO	Echo a message
ECHOERR	Echo a message to stderr
ECHOS	Echo a message with no CR/LF
ECHOSERR	Echo with no CR/LF to stderr
ECHOX	Echo with no expansion
ECHOXERR	Echo with no expansion to stderr
FONT	Change console font
INKEY	Get a keystroke
INPUT	Get an input line
KEYSTACK	Send keystrokes to app
MSGBOX	Popup message box
OSD	Display floating text
PLAYAVI	Play an .AVI file
PLAYSOUND	Play a sound file
PRINT	Print a file
PRINTF	Formatted output
QUERYBOX	Popup input box
SCREEN	Position cursor
SCRPUT	Write directly to screen
SENDHTML	Send HTML email
SENDMAIL	Send email
SMPP	Send SMS message
SNMP	Send SNMP trap

SNPP	Send message to pager
TASKDIALOG	Popup Windows task dialog
VSCRPUT	Write text vertically

Window management commands

	Description
ACTIVATE	Activate or set window state
DESKTOP	Create or switch desktops
POSTMSG	Send a message to a Window
TITLE	Set window title
VDESKTOP	Manage Windows 10 virtual desktops
WINDOW	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	Define or display aliases
BEEP	Beep the speaker
BREAKPOINT	Set a batch debugger breakpoint
CALL	Call another batch file
CANCEL	End batch file processing
COMMENT	Enter multiline comments
DEBUGSTRING	Send text to system debugger
DEFER	Defer a command until the batch file exits
DELAY	Wait for specified time
DO	Batch file looping
ENDLOCAL	Restore a SETLOCAL
EJECTMEDIA	Eject a removable drive
FOR	Repeat a command
FUNCTION	Create or edit user functions
GLOBAL	Run command in subdirectories
GOSUB	Call batch subroutines
GOTO	Go to a batch file label
IF	Conditional command execution
IFF	Conditional command execution
JABBER	Send an IM
LOADBTM	Load batch files as .BTM
LOADMEDIA	Close CD-ROM / DVD drive door
LOCAL	Local variables for batch files and library functions
ON	Batch file error trapping
PAUSE	Wait for input
QUIT	Exit batch file
REM	Remark
REPEAT	Execute counted loop
RETURN	Return from GOSUB

SETLOCAL	Save environment, aliases, and functions
SHIFT	Shift batch file parameters
SWITCH	Batch file switch / case
TEXT	Display text in batch file
TRANSIENT	Toggle shell transient mode
UNALIAS	Remove aliases
UNFUNCTION	Remove user-defined functions
VBEEP	Flash the screen and beep
WEBFORM	Post data to web servers
WEBSOCKET	Connect to WebSocket and send string
WEBUPLOAD	Upload files to web servers

Environment and path commands

	Description
DIRENV	Configure environment per-directory
ESET	Edit variables or aliases
PATH	Set or display PATH
SET	Set or display environment variables
SETP	Set or display environment variables in another process
UNSET	Remove environment variables
UNSETP	Remove environment variables in another process

System configuration and status

	Description
ASSOC	Windows file associations
ASSOCIATE	Combine ASSOC and FTYPE
BREAK	Define or display Ctrl-C state
CHCP	Display or change code page
CLS	Clear the display window
COLOR	Change the display colors
DATE	Display or change date
EVENTLOG	Write to Windows event log
FREE	Display disk space
FTYPE	Display or edit file types
KEYBD	Set keyboard toggles
KEYS	Enable or disable history list
LOG	Save log of commands
MEMORY	Display memory statistics
MOUNTISO	Mount ISO disks
OPTION	Configure the *TCC* console
PLUGIN	Load or unload plugin DLL
PSUBST	Persistent SUBST
REBOOT	Reboot system
RESOLUTION	Change display resolution
RESTOREPOINT	Create / delete / display system restore points

SETDOS	Internal options
SERVICES	Display, stop, or start services
SHORTCUT	Create a Windows shortcut
TASKBAR	Call Windows Taskbar functions
TASKEND	End a task
TASKLIST	Display Windows task list
TIME	Set or display time
UNMOUNTISO	Unmount ISO disk
VERIFY	Display or set disk verification
VER	Display version
VOL	Display or set disk volume label
WAKEONLAN	Send "Wake-On-LAN" packet
WMIQUERY	Query the Windows Management Interface
WMIRUN	Run WMI methods

Monitoring commands

	Description
BTMONITOR	Monitor Bluetooth connections
CLIPMONITOR	Monitor Windows clipboard
DATEMONITOR	Monitor current date and time
DEBUGMONITOR	Monitor OutputDebugString API
DISKMONITOR	Monitor disk usage
EVENTMONITOR	Monitor event log
FIREWIREMONITOR	Monitor FireWire devices
FOLDERMONITOR	Monitor folders and/or files
LOCKMONITOR	Monitor session locking / unlocking
NETMONITOR	Monitor network connections
PROCESSMONITOR	Monitor processes
POWERMONITOR	Monitor system power
REGMONITOR	Monitor Windows registry keys
SCREENMONITOR	Monitor Windows screen saver
SERVICEMONITOR	Monitor Windows services
USBMONITOR	Monitor USB devices

Compression / Decompression commands

	Description
BZIP2	Compress files to bz2 archive
GZIP	Compress files to .gz archive
JAR	Add files to Java jar archive
TAR	Add files to tar archive
UNBZIP2	Extract files from bz2 archive
UNGZIP	Extract files from .gz archive
UNJAR	Extract files from Java jar archive
UNTAR	Extract files from tar archive
UNZIP	Unzip files from archive

ZIP	Zip files to archive
ZIPSFX	Create self-extracting executable
7UNZIP	Extract files from 7Zip archive
7ZIP	Compress files to 7Zip archive

Other commands

	Description
?	Display list of internal commands, or prompt to execute a command
CHRONIC	Run command and hide STDOUT & STDERR
DETACH	Start app detached
EXCEPT	Exclude files from a command
EXIT	Exit *TCC*
EXPR	Evaluate expressions
LIBRARY	Load, display, or delete library functions
LUA	Call the internal Lua interpreter
PEE	Redirect STDOUT to multiple pipes
PSHELL	Execute Powershell script or command
SHRALIAS	Share aliases & functions
SSHEXEC	Connect to remote host and run shell
START	Start a new session
REXEC	Remotely execute command
RSHELL	Remotely execute command
SPONGE	Read STDIN and write to file
TEE	Pipe "tee-fitting"
TIMER	Stopwatch
TS	Timestamp pipe output
WAITFOR	Wait for app exit or input idle
WHICH	Display command information

4.2.3   ?

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. If you have disabled a command with SETDOS /I, 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 command will be executed:

4.2.4 ACTIVATE

Purpose: Activate a window, set its state, or change its title

Format: ACTIVATE [/R] "*title*" [MAX | MIN | RESTORE | DESKTOP | CLOSE | ENABLE | DISABLE | TOPMOST | NOTOPMOST | TOP | BOTTOM | HIDE | FORCEMIN | VDESKTOP=*id* | /FLASH=*type,count* | /ICON=*iconfile* | /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
type	One or more of the following values:
	0 - stop flashing
	1 - flash the window caption
	2 - flash the taskbar button
	4 - flash continuously until WINDOW is called again with the /FLASH type set to 0
	12 - flash continuously until the window comes to the foreground (cannot be used with 4)
count	Number of times to flash the window
top	New location of the top border of the window, in pixels
iconfile	New caption / task bar icon (an .ico file or an executable)
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, TITLE, and WINDOW.

Usage:

ACTIVATE activates, and optionally modifies, another session's window. It is not intended to modify the characteristics of the current TCC-RT session (use TITLE or WINDOW for that purpose).

Title specifies the name of the target window to be activated. You can use wildcards, 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.

If ***title*** begins with a **=**, it is assumed to be a process ID instead of a title. (Note that this is less reliable than providing a title, as a process can have multiple top-level windows.)

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.
DESKTOP	Activates the Windows desktop.
CLOSE	Sends a "close" message to close the window.
ENABLE	Enable mouse and keyboard input.
DISABLE	Disable mouse and keyboard input.
TOPMOST	Keeps the window on top of all other windows until it closes, or **NOTOPMOST** is used.
NOTOPMOST	Allows other windows to overlay the window (this is the normal state for most windows).
TOP	Moves the window to the top of the window order, above all other non-**TOPMOST** windows.
BOTTOM	Moves the window to the bottom of the window order.
HIDE	Makes the window invisible (to make the window visible again, use **RESTORE**).
FLASH	Flash the window.
ICON	Change the window's caption and task bar icon.
POS	Sets the window position and size (in pixels).
TRANS	Transparency level, where n=0 (invisible) to 255 (opaque) (does not work for console windows).
TRAY	Move the specified window to the system tray.
VDESKTOP	Move the window to another virtual desktop. *id* can be either a desktop number (1-n), the GUID for that desktop, or the desktop name. See VDESKTOP for more details.
FORCEMIN	Force the window to be minimized even if the thread that owns the window is not responding.
"newtitle"	Changes the window title.

Numeric input may be entered in either decimal format (a sequence of 0-9 digits) 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 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.

/POS - accepts a * value for any of the arguments. If the value is *, ACTIVATE will use the existing position / width / height value. For example, to resize a window without moving it:

```
ACTIVATE "title" /POS=*,*,1200,800
```

To move a window without resizing it:

```
ACTIVATE "title" /POS=200,400,*,*
```

Examples:

The examples below first maximizes, and then renames the window originally called "Take Command":

```
activate "Take Command" max
activate "Take Command" "Take Command is Great!"
```

4.2.5   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 [/GL /LL /P] [*wildname*]

Definition mode:
ALIAS [/G /GL /LL /O] [/R [/Z] 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)](#)   [/O(verwrite)](#)
[/GL (global list)](#)   [/P(ause)](#)
[/L(ocal)](#)   [/R(ead file)](#)
[/LL (local alias list)](#) [/Z (replace list)](#)

See also: [UNALIAS](#), [ESET](#), and [Aliases](#).

Usage:

- ▶ [Overview](#)
- ▶ [Displaying Aliases](#)
- ▶ [Multiple Commands and Special Characters in Aliases](#)
- ▶ [Nested Aliases](#)
- ▶ [Temporarily Disabling Aliases](#)
- ▶ [Partial (Abbreviated) Alias Names](#)
- ▶ [Keystroke Aliases](#)
- ▶ [Directory Aliases](#)
- ▶ [Saving and Reloading Your Aliases](#)
- ▶ [Alias Parameters](#)
- ▶ [Expanding Aliases at the Prompt](#)
- ▶ [Local and Global Aliases](#)
- ▶ [Retaining Global Aliases with SHRALIAS](#)

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 /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. 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-RT had to search the PATH for it. In addition, PATH can be shorter, which will speed up searches for other programs.

If you apply this technique for each application program, you can often reduce your PATH to just two or three directories containing utility programs, and significantly reduce the time it takes to load most software on your system. Before removing a directory from the PATH, you will need to define aliases for all the executable programs you commonly use which are stored in that directory.

TCC-RT also supports Directory Aliases, a shorthand way of specifying pathnames.

An Alias name can be *contained* in a variable. When TCC-RT does variable expansion on a command line, it will check if the expansion changed the first argument on the line, and if so TCC-RT will check to see if the new argument is an alias. For example:

```
Alias %AliasName=Echo Hello

:: Will output: Echo Hello
Alias MyAlias

:: Will output: Hello
%MyAlias
```

Aliases are stored in memory, and are not saved automatically when you turn off your computer or end your current TCC-RT 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](#) 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](#) 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](#) and may also use the [escape character](#).

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.

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](#) or [IFF](#) command is used to prevent an infinite loop. The two aliases below demonstrate alias nesting (one alias invoking another). The first line defines an alias which runs in the current directory, and executes **Word** located in the **E:\WINWORD**. The second alias changes directories with the [PUSHD](#) command, runs the **WP** alias, and then returns to the original directory with the [POPD](#) 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-RT not to attempt to interpret that command as another (nested) alias. An asterisk used this way must be preceded by a space or the command separator and followed immediately by an internal or external command name.

By using an asterisk, you can redefine the default options for any internal or external command. For example, suppose that you always want to use the DIR command with the **/2** (two column) and **/P** (pause at the end of each page) options:

```
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-RT would repeatedly re invoke the **DIR** alias, rather than running the DIR command. This would cause an "Alias loop" or "Command line too long" error. The asterisk forces interpretation of the second DIR as a command, not an alias.

An asterisk also helps you keep the names of internal commands from conflicting with the names of external programs. For example, suppose you have a program called *DESCRIBE.EXE.* Normally, the internal DESCRIBE command will run anytime you type DESCRIBE. But two simple aliases will give you access to both the *DESCRIBE.EXE* program and the DESCRIBE command:

```
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 command. The second line defines **FILEDESC** as a new name for the internal DESCRIBE command. The asterisk is needed in the second command to indicate that the following word means the internal command DESCRIBE, not the **describe** alias which runs your external program.

Another way to understand the asterisk is to remember that a command is always checked for an alias first, then for an internal or external command, or a batch file. The asterisk at the beginning of a command name simply skips over the usual check for aliases when processing that command, and allows TCC-RT 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 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 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 and autoexecute.

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-RT 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-RT 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](#) 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 key directive option (by default, the **Esc** key, 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](#) command in the alias value.

Be careful if you assign aliases to keys that are already used at the command line. The keystroke alias definitions take precedence, so they will disable the matching command line editing key.

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](#) for a complete listing of key names and a description of the key name format.

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-RT has no way of knowing what is a valid argument for external applications.

Directory alias names can be either two or more alphanumeric characters followed by a colon, or a single digit followed by a colon. You cannot [abbreviate](#) directory aliases.

Directory aliases support environment variable expansion.

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-RT 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-RT.

When you define aliases in a file that will be read with the **ALIAS /R** command, do not use 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 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 for details. (Alias examples in this section assume the TCC-RT 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$** has a special meaning. TCC-RT interprets it to mean "the entire command line, from parameter *n* to the end." If *n* is not specified, it has a default value of **1**, so **%$** 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 command to redefine the defaults for SET. It should be entered on one line:

```
alias set = `iff %# == 0 then & *set /p & else & *set %$ & endiff`
```

This modifies the SET command so that if SET is entered with no parameters, it is replaced by SET /P (pause after displaying each page), but if SET 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-W** 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-W 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 local and/or global lists. The selection is made during TCC-RT startup, using the **/L** or **/LA** START or startup options, or by the Local Aliases and Global Aliases configuration options, or interactively with the **ALIAS /G, /GL, /L, and /LL** options. The global alias list is limited to 256 K 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-RT. They will not be visible in other shells or other sessions.

With a global alias list, all copies of TCC-RT, 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-RT.

If you don't specify /GL or /LL, TCC-RT will first look for aliases in the local list. If there is no local list or the alias is not found, TCC-RT will search the global list (if it exists).

There is no fixed rule for determining whether to use local or global alias lists. 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 / ENDLOCAL inside a batch file, changes in local alias definitions (or in global definitions if you don't have a local alias list) are restored by the ENDLOCAL.

Retaining Global Aliases with SHRALIAS

If you select a global alias list for TCC-RT you can share the aliases among all running copies of TCC-RT. When you close all TCC-RT sessions, the memory for the global alias list is released, and a new, empty alias list is created the next time you start TCC-RT.

If you want the alias list to be retained in memory even when no TCC-RT session is running, you need to execute the SHRALIAS command, which performs this service for the global alias list, the global user-defined functions list, the global command history list, and the global directory history list. You may find it convenient to execute SHRALIAS from your TCSTART file.

SHRALIAS retains the global alias list in memory, but cannot preserve it when Windows itself is shut down. To save your aliases when restarting Windows, you must store them in a file and reload them after the system restarts. For details on how to do so, see Saving and Reloading Your Aliases above.

The CD_ENTER and CD_LEAVE Aliases

When changing directories, TCC-RT will look for (and execute if found) the following aliases:

CD_Leave - TCC-RT will execute this alias when it is about to change the current directory. TCC-RT will pass the name of the current directory (%1) and the name of the new directory (%2).

CD_Enter - *TCC-RT* will execute this alias immediately after changing the current directory. *TCC-RT* will pass the name of the new directory (%1).

These aliases let you customize your environment based on the current directory.

The UNKNOWN_CMD Alias

If you create an alias with the name **UNKNOWN_CMD**, it will be executed any time TCC-RT 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-RT will loop up to 10 times, then display the **UNKNOWN_CMD loop** error.

If an **UNKNOWN_CMD** alias does not exist, TCC-RT will search the plugins for an UNKNOWN_CMD command and execute it if found.

Warnings

When you define an alias in the command line (i.e., without using the /R 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. If you already have a global alias list (for example, in another TCC-RT instance or in SHRALIAS), ALIAS will not do the conversion. The /G must be the only argument.
/GL	Read from and write to the global alias list. If you have both local and global alias lists defined and do not specify /GL, ALIAS will default to using the local list.
/L	Switch from a global to a local alias list. If you already have a local alias list, ALIAS will not do the conversion. The /L must be the only argument.
/LL	Read from and write to the local alias list.
/O	Don't overwrite existing values (only valid in combination with /R).
/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).
/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***. Do not use back quotes around the value with /R. 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
```

ALIAS /R definitions can span multiple lines in the file if each line of the definition, except the last, is terminated with an escape character.

ALIAS /R will read from stdin if no filename is specified and input is redirected:

```
alias /r <
```

| **/Z** | Overwrite the existing alias list with the contents of the specified file. Can only be used with a single /R file argument. ALIAS /R /Z is 20x faster than an ALIAS /R, because it doesn't have to check for existing aliases and append new aliases to the end of the list. Do not use single back quotes around your alias arguments with /Z. |

4.2.6   ASSOC

| ***Purpose:*** | Modify or display relationships between file extensions and file types stored in the Windows registry |

| ***Format:*** | ASSOC [/P /R [*file*...] | [.*ext*[=[*filetype*]] /U] |

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)   /U(ser)
/R(ead)

See also: FTYPE, ASSOCIATE, and Executable Extensions.

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. 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 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. |

| **/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 if no filename is specified and input is redirected.

/U   Display or set the association in HKCU\Software\Classes.

4.2.7   ASSOCIATE

Purpose: Display, create, or delete file / command associations.

Format: ASSOCIATE [/D /F /P[*n*] /R [*file*...] /U /V:*verb* [*.ext*[=[*command*]]]

file	One or more input files to read association definitions from.
.ext	The file extension whose associated command you want to display or set.
command	The executable command to run for the specified file extension

[/D(elete)](#)   [/R(ead)](#)
[/F(orce)](#)   [/U(ser)](#)
[/P(ause)](#)   [/V(erb)](#)

See also: [ASSOC](#), [FTYPE](#), and [Executable Extensions](#).

Usage:

You must be running in an elevated session to use ASSOCIATE (unless you're using the /U option).

ASSOCIATE combines the ASSOC and FTYPE commands. It allows you to create, modify, or display associations between file extensions and commands types stored in the Windows registry.

ASSOCIATE manages Windows file associations stored under the registry handle HKEY_CLASSES_ROOT (or HKEY_CLASSES_USER), and discussed in more detail under [Windows File Associations](#). If you are not familiar with file associations be sure to read about them before using ASSOCIATE.

If you invoke ASSOCIATE with no parameters, it will display the current associations. If you include a *.ext*, with no equal sign or *command*, ASSOCIATE will display the command associated with that extension.

If you include the equal sign and *command*, ASSOCIATE will create or update the association for extension *.ext* to refer to the specified command.

ASSOCIATE 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:

/D   Delete the association for the specified *.ext.*

/F   Force an overwrite of an existing association.

/P[*n*] 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. (Only useful when running ASSOCIATE with no arguments.) The /P option has an optional argument *n* that specifies the number of seconds to wait for a keystroke before continuing.

/R This option loads an association list from a file. The format of the file is the same as that of the ASSOCIATE display:

        .ext=command

where *.ext* is an extension, which is to be associated with *command*.

You can load multiple files with one ASSOCIATE /R command by placing the names on the command line, separated by spaces:

        associate /r assoc1.lst assoc2.lst

You can insert comments in the file by prefixing the line with a colon (**:**).

ASSOCIATE /R will read from stdin if no filename is specified and input is redirected.

/U Display or set the association in HKCU\Software\Classes.

/V:*verb* ASSOCIATE defaults to reading and writing to SHELL\OPEN\COMMAND. You can use a different verb by specifying the /V option. For example, to create a PRINT verb for .TXT files:

        ASSOCIATE /V:PRINT .txt=%%SystemRoot%%
        \system32\NOTEPAD.EXE /p %%1

If you use * for the verb, ASSOCIATE will display all of the shell verbs and their commands for the specified extension.

4.2.8 ATTRIB

Purpose: Change or view file and subdirectory attributes

Format: ATTRIB [/A:[[-+]rhsa] /D /E /I"*text*" /L /N[EJ] /O:[-]acdeginorstuz /P[*n*] /Q /S[[+]n]] [+|-[AHIOPRSTUVX]] [@*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 for details).

/A: (Attribute select)	/N(o)
/D(irectories)	/O:...(order)
/E (No error messages)	/P(ause)
/I"text" (match description)	/Q(uiet)
/L (symbolic Link)	/S(ubdirectories)

Attribute flags:

Clear	Set	Attribute affected
-A	+A	archive

-C	+C	compressed
-H	+H	hidden
-I	+I	not content indexed
-O	+O	offline
-P	+P	pinned (Windows 10 and OneDrive only)
-R	+R	read-only
-S	+S	system
-T	+T	temporary
-U	+U	unpinned (Windows 10 and OneDrive only)
-V	+V	integrity (Windows Server 2012R2+ ReFS only)
-X	+X	no_scrub_data (Windows Server 2012R2+ ReFS only)

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.

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.

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 **[+|-[AHIOPRSTUVX]]** part of the format), or with the DIR /T command.

The primary use of ATTRIB is to set attributes. For example, you can set the read-only and hidden attributes for the file *MEMO*:

```
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.EXE*:

```
attrib +a *.txt +s −a test.exe
```

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** (virtualized), **E** (encrypted), **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 **[+|-[ADHIOPRSTUVX]]** 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 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*
```

ATTRIB sets three internal variables:

%_attrib_dirs	The number of directories modified
%_attrib_files	The number of files modified
%_attrib_errors	The number of errors

Options:

/A: Select only those files that have the specified attribute(s) set. See Attribute Switches 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. See @file lists for details.

You can specify **/A:=** to display a dialog to help you set individual attributes.

/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.   .

/l"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

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. See @file lists for details.

/L	Set or display the attributes of the symbolic link versus the target of the symbolic link.
/N	Do everything except actually change the attributes. This option is useful for testing what the result of a complex ATTRIB command will be.

A **/N** with one of the following arguments has an alternate meaning:

e	Don't display errors.
j	Skip junctions (when used with /S)

/O:...	Sort the files before processing.

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, instead of numerically when numeric substrings are included in the name or extension.
c	Sort by compression ratio
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 description
o	Sort by owner
r	Reverse the sort order for all options
s	Sort by size
t	Same as **d**
u	Unsorted
z	Same as **s**

/P[*n*]	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. The /P option has an optional argument *n* that specifies the number of seconds to wait for a keystroke before continuing.
/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 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.

If you specify a **+** followed by a number after the /S, ATTRIB will not modify any file attributes until it gets to that depth in the subdirectory tree. For example, if you have a directory tree \a\b\c\d\e, /S+2 will not modify anything in \a or \a\b.

4.2.9   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.

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 64-bit versions of Windows do not support playing sounds through the Windows Beep API, TCC-RT uses DirectSound for BEEP.

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 (a sequence of 0-9 digits) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

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

Example:

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...
```

4.2.10 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-RT (though not necessarily in child processes). In CMD, BREAK OFF doesn't actually do anything, so setting it in TCC-RT will introduce a possible incompatibility with existing batch files.

4.2.11 BREAKPOINT

Purpose: Set a batch debugger breakpoint on the current line

Format: BREAKPOINT

Usage:

BREAKPOINT sets a breakpoint on the current line, stopping a "Step Out" sequence. If the batch debugger is not active, BREAKPOINT is ignored.

4.2.12 BTMONITOR

Purpose: Monitor when a Bluetooth device is connected or disconnected.

Format: BTMONITOR [name /C]
                   BTMONITOR [/=] *name* [Connected | Disconnected] [*n* | FOREVER] *command*

              [/C(lear)](#)

Usage:

The Bluetooth name can include wildcards.

The command line will be parsed and expanded before BTMONITOR 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 the last argument on the line is a single **(,** it is interpreted as the beginning of a command group. BTMONITOR will append the following lines (in a batch file) or prompt you for more input (at the command line) until it gets a closing **).**

If you don't enter any arguments, BTMONITOR will display the Bluetooth 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-RT thread is also performing I/O. You may need to use START or DETACH in **command** to avoid conflicts.

BTMONITOR creates several environment variables when a Bluetooth device is connected or disconnected that can be queried by **command**. The variables are deleted after **command** is executed.

_btindex	The index of the Bluetooth device being connected or disconnected (for the @btdevice... functions)
_btaddress	The address of the Bluetooth device being connected or disconnected
_btname	The name of the Bluetooth device being connected or disconnected

Options:

/C	Remove the Bluetooth monitor.

4.2.13 BZIP2

Purpose: Create bzip2 (*.bz2) compressed archives

Format: BZIP2 [/= /A:[[-][+]rhsdaecjot] /A /M /Q /V] *bzip2archive* [@*file*] *file*

bzip2archive The .bz2 file to work with
file   The file to compress

/A:... (attribute switch)	M(ove)
/A(dd)	/Q(uiet)
/C(ontents)	/V(iew)

See also: UNBZIP2.

Usage:

BZIP2 is normally used for compressing a single file; if you need to compress multiple files you should use the ZIP (or TAR) command.

You can specify a pathname for *bzip2archive*. If you don't provide an extension, and the filename as entered doesn't exist, BZIP2 adds ".bz2". If you don't specify an operation, BZIP2 will default to Add.

Options:

/=	Display the BZIP2 command dialog to help you set the command line options. The /= option can be anywhere on the line; additional options will set the appropriate fields in the command dialog.
/A:...	Select only those files that have the specified attribute(s) set. See Attribute Switches for information on the attributes which can follow /A:. Do not use /A: with **@*file*** lists. See @file lists for details.
	You can specify **/A:=** to display a dialog to help you set individual attributes.
/A	Add the specified file to the archive. (This is the default.)
/C	Display (on standard output) the contents of a file in the zip archive.

/M	Delete the file from the disk after adding them to the bzip2 file.
/Q	Don't display the file being compressed.
/V	View the contents of the .bz2 file (date, time, and filename).

4.2.14 CALL

Purpose: Execute one batch file from within another.

Format: CALL *file | :label* [*p1* [*p2* ...]]

file	The batch file to execute.
:label	A label in the current batch file.
p1, p2,...	Parameters for the batch file or subroutine

See also: <u>CANCEL</u> and <u>QUIT</u>.

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 <u>CANCEL</u> 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 <u>QUIT</u> 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 64 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 command), or it should terminate batch file processing with CANCEL. Do not restart or CALL the original batch file from within the called file as this may cause an infinite loop or a stack overflow.

Calling a label

To provide compatibility with CMD, which does not support the GOSUB command for subroutines in the same batch file, you may create a subroutine starting with a label and terminated by any of the following:

- the end of the batch file
- QUIT
- EXIT
- CANCEL

Note that the last two do NOT return control to the CALL command. Do not use the RETURN 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 conditional commands (**&&** and **||**).

See also GOSUB and user-defined functions.

4.2.15 CALLER

Purpose:	Return the context of the current batch call

Format:	CALLER [*n*]

Usage:

Returns the context of the current batch call, including the line number, (optional) subroutine label, and batch file or library name.

If *n* is not specified, CALLER displays the line number and batch file or library name for the current line. If *n* is specified, CALLER displays the line number, subroutine label (or "main" if not in a subroutine), and batch or library name. The *n* value for the current line is *0,* the previous call in the program stack is *1, etc.*

4.2.16 CANCEL

Purpose: Terminate batch file processing

Format: CANCEL [*value*]

> **value** The numeric exit code to return to TCC-RT.

See also: CALL and QUIT.

Usage:

The CANCEL command ends all batch file processing, regardless of the batch file nesting level. Use QUIT to end a nested batch file and return to the previous batch file.

You can CANCEL at any point in a batch file. If CANCEL is used from within an alias it will end execution of both the alias and any batch files which are running at the time.

The following batch file fragment compares an input line to "end" and terminates all batch file processing if it matches:

```
input Enter your choice:  %%option
if "%option" == "end" cancel
```

If you specify a ***value***, CANCEL will set the ERRORLEVEL or exit code to that value (see the IF command, and the %? variable).

4.2.17 CAPTURE

Purpose: CAPTURE does video and / or audio screen capture. It supports H264, H265, VP80, VP90, MP3, FLAC, and AAC.

Format: CAPTURE
"filename" [/Start=n /End=n /FPS=n /HWND=n /Monitor=n /Rect=top,left,bottom,right /Video=[H264 | HEVC | VP80 | VP90] /AudioFormat=[MP3 | AAC | FLAC] /AudioFrom="name" /Threads=n /C /E /P]..

"filename" - The output filename (.mp4 or .asf for video; .mp3, .aac, .flac for audio)

/AudioFormat (Audio encoding format)	/Monitor (Monitor to capture)
/AudioFrom - (Audio source)	/P(ause capture)
/C(apture cursor)	/RECT (Window rectangle to capture)
/E(nd capture)	/Start (Start time)
/End (End time)	/Threads (Video encoding threads)
/FPS (Frames per second	/Video (Video encoding format)
/HWND (Window to capture)	

Usage:

If you do not specify /End, CAPTURE will continue capturing the screen until you call it again with the /E option.

If you do not specify /HWND or /RECT, CAPTURE will capture the desktop.

CAPTURE runs in a separate thread, so it will not block the current *TCC / Take Command* window.

Options:

/C	Capture the cursor activity. The default is to not save cursor movements.
/E	End the capture. If you do not specify /End, CAPTURE will continue capturing video and/or audio until you do a CAPTURE /E.
/P	Pause the capture.
/Start	The start time in seconds (default 0).
/End	The end time in seconds. If you do not specify /End, you will need to run CAPTURE /E to stop the capture.
/FPS	Frames per second for video capture (default 25)
/HWND	The window handle to capture
/Monitor	The monitor to capture (1 - n)
/Rect	The window rectangle to capture
/Threads	The number of threads for the video encoding (default 1, maximum 16)
/Video	Video encoding format (H264, HEVC, VP80, or VP90)
/AudioFormat	Audio encoding format (MP3, AAC, FLAC)
/AudioFrom	The friendly name of the audio source. You can use wildcards in the name; for example: /AudioFrom="HD Audio*"

4.2.18 CD / CHDIR

Purpose: Display or change the current directory

Format: CD [/D /N /R /X] [*path* | - | *:path*]

path	The directory to change to, optionally including a drive letter
folder	A Windows Shell folder name

/D(rive)   /R(eparse point)
/N(o extended search)   /X (exclude)

See also: CDD, MD, PUSHD, and RD.

Internet: Can be used with FTP Servers.

Usage:

CD and CHDIR are synonyms. You can use either one.

CD lets you navigate through a drive's subdirectory structure by changing the current working directory. If you enter CD and a directory name, the named directory becomes the new current directory. For example, to change to the subdirectory *C:\FINANCE\MYFILES*:

```
[c:\] cd \finance\myfiles
[c:\finance\myfiles]
```

Every disk drive on the system has its own current directory. Specifying both a drive and a directory in the CD command will change the current directory on the specified drive, but will not change the default drive (unless you use the /D 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 option, or preferably the CDD 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 drive), you must enclose it in double quotes.

If *path* begins with a **~** (tilde), CD will substitute to the user's home directory, as defined by HOME in the environment. (If HOME doesn't exist, TCC-RT will look for %HOMEDRIVE + HOMEPATH.)

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). 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 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 *path* to force an extended directory search.

If the EverythingSearch option is set, CD will use **Everything Search** (http://www.voidtools.com) instead of JPSTREE.IDX for fuzzy directory searches. Everything Search is faster, but will only work on local NTFS drives. Setting EverythingSearch is the equivalent of setting FuzzyCD=3 (***name***). The *Take Command* installer will install **Everything Search** automatically.

CD saves the current directory before changing to a new directory. You can switch back to the previous directory by entering CD -. (There must be a space between the CD command and the hyphen.) You can switch back and forth between two directories by repeatedly entering CD -. The saved directory is the same for both the CD and CDD commands. Drive changes and automatic directory changes also modify the saved directory, so you can use CD - to return to a directory that you exited with an automatic directory change. TCC-RT recognizes a single hyphen on the command line as an internal alias for **CDD -**.

You can also use CD to display or change the current directory on an FTP server opened with IFTP. For example:

```
cd ftp:
ftp://ftp.microsoft.com/
```

```
cd ftp:/pub
```

CD never changes the default drive, unless the /D 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.

At startup, TCC-RT saves the last directory from SHRALIAS or (if loaded by TCSTART) the directory history list to the "CD -" buffer.

You can also CD to one of the predefined Windows folders. The syntax is:

```
CDD :foldername
```

where *foldername* can be:

```
AccountPictures
AddNewProgramsFolder
AdministrativeTools
AppData
ApplicationShortcuts
AppsFolder
AppUpdatesFolder
Cache
CameraRoll
CDBurning
ChangeRemoveProgramsFolder
CommonAdminTools
CommonAppData
CommonDesktop
CommonDocuments
CommonDownloads
CommonMusic
CommonPictures
CommonPrograms
CommonRingtones
CommonStartMenu
CommonStartup
CommonTemplates
CommonVideo
ConflictFolder
ConnectionsFolder
Contacts
ControlPanelFolder
Cookies
Cookies\Low
CredentialManager
CryptoKeys
Desktop
DeviceMetadataStore
```

```
DocumentsLibrary
Downloads
dpapiKeys
Favorites
Fonts
Games
GameTasks
History
HomeGroupCurrentUserFolder
HomeGroupFolder
ImplicitAppShortcuts
InternetFolder
Libraries
Links
LocalAppData
LocalAppDataLow
MusicLibrary
MyComputerFolder
MyMusic
MyPictures
MyVideo
Nethood
NetworkPlacesFolder
OneDrive
OneDriveCameraRoll
OneDriveDocuments
OneDriveMusic
OneDrivePictures
Personal
PicturesLibrary
PrintersFolder
PrintHood
Profile
ProgramFiles
ProgramFilesCommon
ProgramFilesCommonX64
ProgramFilesCommonX86
ProgramFilesX64
ProgramFilesX86
Programs
Public
PublicAccountPictures
PublicGameTasks
PublicLibraries
QuickLaunch
Recent
RecordedTVLibrary
RecycleBinrFolder
```

```
ResourceDir
RingTones
RoamedTileImages
RoamingTiles
SavedGames
Screenshots
Searches
SearchHistoryFolder
SearchHomeFolder
SearchTemplatesFolder
SendTo
StartMenuStart Menu
Startup
SyncCenterFolder
SyncResultsFolder
SyncSetupFolder
System
SystemCertificates
SystemX86
Templates
ThisPCDesktopFolder
UsersFilesFolder
UserPinned
UserProfiles
UserProgramFiles
UserProgramFilesCommon
UsersLibrariesFolder
VideosLibrary
Windows
```

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. In most cases you should use CDD, which performs the same function.

/N   Skips the standard extended directory search 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.

/R   Change to the target of the reparse point (hard or symbolic link).

/X   Don't save the current directory to the Directory History list..

4.2.19 CDD

Purpose: Change the current disk drive and directory

Format: CDD [/A /D[*drive* ...] /N[J] /S[n][*drive* ...] /U[n][*drive* ...] /X] [*path* | - | *:folder*]

path	The name of the directory (or drive and directory) to change to.
folder	A Windows Shell folder name
drive	A drive or list of drives to include in the extended directory search database.

/A(ll drives)	/S(earch tree)
/D(elete from JPSTREE.IDX)	/T (Also change Folders directory)
/N(o extended search)	/TO Only change Folders directory)
/NJ (Skip junctions)	//U(pdate tree)
/R(eparse point)	/X (exclude from directory history)

See also: CD, MD, PUSHD, and RD.

Usage:

CDD is similar to the CD 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.

If *path* begins with a **~** (tilde), CD will substitute to the user's home directory, as defined by HOME in the environment. (If HOME doesn't exist, TCC-RT will look for %HOMEDRIVE + HOMEPATH.)

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 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, then the extended directory search 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.

If the EverythingSearch option is set, CDD will use **Everything Search** (http://www.voidtools.com) instead of JPSTREE.IDX for fuzzy directory searches. Everything Search is slightly faster, but will only work on local NTFS drives. Setting EverythingSearch is the equivalent of setting FuzzyCD=3 (***name***). The *Take Command* installer will install **Everything Search** automatically.

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 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. TCC-RT recognizes a single hyphen on the command line as an internal alias for **CDD -**.

At startup, TCC-RT saves the last directory from SHRALIAS or (if loaded by TCSTART) the directory history list to the "CD -" buffer.

Windows limits the permissible length of the full subdirectory name (see the Directories and Subdirectories topic for information on directory names).

When changing directories, TCC-RT 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.

You can also CD to one of the predefined Windows shell folders. The syntax is:

```
CDD :foldername
```

where *foldername* can be:

```
        AccountPictures
        AddNewProgramsFolder
        AdministrativeTools
        AppData
        ApplicationShortcuts
        AppsFolder
        AppUpdatesFolder
        Cache
        CameraRoll
        CDBurning
        ChangeRemoveProgramsFolder
        CommonAdminTools
        CommonAppData
        CommonDesktop
        CommonDocuments
        CommonDownloads
        CommonMusic
        CommonPictures
        CommonPrograms
        CommonRingtones
        CommonStartMenu
        CommonStartup
        CommonTemplates
        CommonVideo
        ConflictFolder
        ConnectionsFolder
        Contacts
        ControlPanelFolder
        Cookies
        Cookies\Low
        CredentialManager
        CryptoKeys
        Desktop
        DeviceMetadataStore
        DocumentsLibrary
        Downloads
```

```
dpapiKeys
Favorites
Fonts
Games
GameTasks
History
HomeGroupCurrentUserFolder
HomeGroupFolder
ImplicitAppShortcuts
InternetFolder
Libraries
Links
LocalAppData
LocalAppDataLow
MusicLibrary
MyComputerFolder
MyMusic
MyPictures
MyVideo
Nethood
NetworkPlacesFolder
OneDrive
OneDriveCameraRoll
OneDriveDocuments
OneDriveMusic
OneDrivePictures
Personal
PicturesLibrary
PrintersFolder
PrintHood
Profile
ProgramFiles
ProgramFilesCommon
ProgramFilesCommonX64
ProgramFilesCommonX86
ProgramFilesX64
ProgramFilesX86
Programs
Public
PublicAccountPictures
PublicGameTasks
PublicLibraries
QuickLaunch
Recent
RecordedTVLibrary
RecycleBinrFolder
ResourceDir
RingTones
```

```
RoamedTileImages
RoamingTiles
SavedGames
Screenshots
Searches
SearchHistoryFolder
SearchHomeFolder
SearchTemplatesFolder
SendTo
StartMenuStart Menu
Startup
SyncCenterFolder
SyncResultsFolder
SyncSetupFolder
System
SystemCertificates
SystemX86
Templates
ThisPCDesktopFolder
UsersFilesFolder
UserPinned
UserProfiles
UserProgramFiles
UserProgramFilesCommon
UsersLibrariesFolder
VideosLibrary
Windows
```

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 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 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).

/R   Change to the target of the reparse point (hard or symbolic link).

/S   Builds or rebuilds the Extended Directory Search 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, and E, and the sharename \\server\dir1, use this command:

```
cdd /s c:\ d:\ e:\ \\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 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* File Explorer window.

/TO   Change the current directory in the *Take Command* File Explorer window without changing the *TCC-RT* current directory.

/U   Updates the Extended Directory Search 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:\
```

If you specify a number after the /U, 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 update the "a", "b", and "c" directories.

Note: The TREEEXCLUDE variable can be used to specify which drives and directories should be ignored when updating the directory database.

/X   Don't save the current directory to the Directory History list.

4.2.20 CHCP

Purpose: Display or change the current system code page

Format: CHCP [/I /P[*n*] /S] [*n*]

             n   A system code page number.

             [/I(nstalled)](#)

[/P(ause)](#)
[/S(upported)](#)

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.

Options:

/I   Show all installed code pages.

/P[*n*]   Pause after each page (only valid with /I or /S). Your options at the prompt are explained in detail under [Page and File Prompts](#). The /P option has an optional argument *n* that specifies the number of seconds to wait for a keystroke before continuing.

/S   Show all supported code pages.

4.2.21 CHRONIC

Purpose: Run a command and hides its STDOUT and STDERR output unless the command fails

Format: CHRONIC [/R] *command ...*

[/R (stderr)](#)

Usage:

CHRONIC runs a command and hides its STDOUT and STDERR output unless the command fails (i.e., the return value is != 0). If the command succeeds, no output is displayed.

CHRONIC will display the STDOUT and STDERR output separately. For example:

```
c:\> CHRONIC testcommand
Exit code: 2
STDOUT:
stdout output here ...
STDERR:
stderr output here ...
```

CHRONIC will only display the STDOUT: / STDERR: labels if the command actually wrote to the matching output.

CHRONIC will only display the STDOUT: / STDERR: labels if the command actually wrote to the matching output.

Options:

/R   Display the output if the command writes to STDERR. If /R is not specified, CHRONIC will only display the output if the command returns a non-zero exit code.

4.2.22 CLIP

Purpose: CLIP displays or modifies the 10 clipboards available in TCC-RT (CLIP0: - CLIP9:)

Format: CLIP [/C *clipn:]* [/R *n]* [/S *clipn: text]* [/Z]

/C - Clear
/R - Rotate
/S - Set clipboard text
/Z - Clear all

Usage:

TCC-RT supports multiple clipboards. They are numbered from CLIP0: - CLIP9:. You can still use CLIP: - it is equivalent to CLIP0:. Clipboards 1 - 9 are only accessible to TCC-RT internal commands and variable functions. External applications will only be able to access the default Windows clipboard (CLIP: / CLIP0:).

The /C and /S options accept the clipboard argument either as a digit (0 - 9) or a string (CLIP0: - CLIP9:).

When an app saves something to the default clipboard (CLIP: or CLIP0:), TCC-RT will rotate the existing clipboard entries before saving the new CLIP0. CLIP0: will become CLIP1:, CLIP1: becomes CLIP2:, etc. The old CLIP9: will be lost. If you save something to CLIP1: - CLIP9:, none of the other clipboard entries will be modified.

If you don't specify a clipboard number (i.e., "CLIP:", CLIP will default to CLIP0:.

The /C and /S options accept either CLIPn: (i.e., CLIP0: to CLIP9:), or just the digit 0 - 9.

If you don't specify any arguments, CLIP will display the current contents of CLIP0: - CLIP9:.

Options:

/C   Clears clipboard *n.*

/R   Rotates the clipboards to make clipboard *n* the default (i.e., CLIP: / CLIP0:).

/S   Sets clipboard *n* to *text*

/Z   Clears all the clipboards

4.2.23 CLIPMONITOR

Purpose:	Monitor changes in the Windows clipboard

Format:	CLIPMONITOR [/C]
	CLIPMONITOR *n command*

n	Number of repetitions (or **FOREVER**)
command	Command to execute when the clipboard is modified

[/C(lear)](#)

Usage:

If you don't enter any arguments, if CLIPMONITOR is active it will display the repeat count and the command.

The command line will be parsed and expanded before CLIPMONITOR 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 the last argument on the line is a single **(,** it is interpreted as the beginning of a command group. CLIPMONITOR will append the following lines (in a batch file) or prompt you for more input (at the command line) until it gets a closing **).**

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-RT thread is also performing I/O. You may need to use [START](#) or [DETACH](#) in ***command*** to avoid conflicts.

Options:

/C	Remove the clipboard monitor.

4.2.24 CLS

Purpose:	Clear the window and move the cursor to the upper left corner; optionally change the default display colors

Format:	CLS [/C /S] [[BRIght] *fg* ON [BRIght] *bg*

fg	The new foreground color
bg	The new background color

[/C(lear buffer)](#)   [/S(croll buffer)](#)

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 scrollback 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 for details about colors.

Options:

/C   Clears the entire scrollback 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 scrollback buffer if it is larger than the visible window. This switch may not give the expected results when the buffer size is less than twice the window size.

4.2.25 COLOR

Purpose: Change the default display colors

Format: COLOR [/F *filename*] [BRIght] *fg* ON [BRIght] bg
COLOR [/FG:*rgb* /BG:*rgb*]
COLOR [/FT:*color* /BG:*color*]
COLOR /P[*color*]

fg   The new foreground color
bg   The new background color

See also: CLS and Colors and Color Names 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-RT also supports the CMD ```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 do not specify a new foreground and background color, COLOR will revert the display colors to those used when TCC-RT was started (for compatibility with CMD).

If you have ANSI enabled and StdColors and/or InputColors set, they will override a COLOR command.

COLOR now supports changing the console color palette with either an .INI file (for example, as used by the ColorTool utility), or an .ITERMCOLORS file. The syntax is:

```
COLOR /F filename
```

If you are running in a Take Command tab window, COLOR will pass the new colors to *Take Command* to update the tab window. You can have a different color palette in each tab window.

You can set the foreground and background color in a TCC-RT console window (not a TCMD tab window) to 16 million or 256 colors with the /FG and /BG options. You must be running Windows 10 and have ANSI enabled.

Options:

/FG:*rgb*	Sets the foreground color to the 16 million color RGB value specified. Valid ranges for r, g, and b are 0-255.
/BG:*rgb*	Sets the background color to the 16 million color RGB value specified. Valid ranges for r, g, and b are 0-255.
/FG:*color*	Sets the foreground color to the 256 color (xterm) value specified. Valid range for *color* is 0 - 255.
/FG:*color*	Sets the background color to the 256 color (xterm) value specified. Valid range for *color* is 0 - 255.
/P	Displays a color picker dialog to select a color. Must be used with /FG or /BG, and cannot be combined with /F.

4.2.26 COMMANDS

Purpose: Display, enable, or disable the TCC-RT internal commands

Format: COMMANDS [/D /E /P] *commandname ...*

Usage:

If you do not enter any arguments, COMMANDS will display all of the internal commands. Disabled commands will be enclosed in parentheses. If you enter command names without a /D or /E, COMMANDS will show the current state of those commands.

Example:

To disable the internal WINSTATION command:

```
commands /d winstation
```

Options:

/D	Disable one or more commands. If you do not provide any command names, COMMANDS will display all of the disabled commands.
/E	Enable one or more commands. If you do not provide any command names, COMMANDS will display all of the enabled commands.
/P	Pause after displaying each page.

4.2.27 COMMENT

Purpose: Mark a block of comments in a batch file

Format: COMMENT
.
.
.
ENDCOMMENT

Usage:

COMMENT can only be used in batch files. Both COMMENT and ENDCOMMENT must be entered as the only commands on their respective lines, and cannot be included in a command group.

The COMMENT command is useful for entering multiline comments without having to prefix each line with a REM.

The lines between COMMENT and ENDCOMMENT are not parsed. As a consequence, no environment variable expansion or other processing is performed. This makes it easy to include special characters, e.g., **< | >** in the text.

4.2.28 COPY

Purpose: Copy data between disks, directories, files, or physical hardware devices (such as your printer or serial port)

Format: COPY [/= /I"*text*"] [/A:... /BAK /C /CF /CRC:type:*filename* /D /DD /DS:[acwu]yyyy-mm-d /E /F /FTP:A /G /GZ /H /J /K /L /M /MD /N[dejnrstz] /O /O:[-]acdeginorstuz /P /Q /R /RCT /S[[+]n] /SX /T /TS[acwu] hh:mm:ss.ms /U /UF /V[n] /W /WAIT=*n* /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 for details)

/A(SCII) copy	/M(odified files
/A:... (Attribute select)	MD (Create target directory)
/B(inary copy)	/N (Disable)
/BAK (backup)	/O(nly if no target)
/C(hanged source files)	/O:... (order)
/CDA (copy directory attributes)	/P(rompt)
/CF (changed 2s+ resolution)	/Q(uiet)
/CRC (create CRC for each file)	/R(eplace)
/D (Copy encrypted files)	/RCT (request compress)
/DD (delete empty directories)	/S(ubdirectories)
/DS (date stamp)	/SX (single target directory)
/E (No error messages)	/T(otals)
/F (No empty subdirectories)	/TS (timestamp)
/FTP:A (ASCII copy)	/U(pdate target)
/G (Display percentage)	/UF (update 2s+ resolution)
/GZ (gzip HTTP files)	/V(erify)

/H (Include hidden files)/G   /W (one-way sync)
/I"text" (Match description)   /WAIT=*n*
/J (Restartable)   /X (Clear archive)
/K (Keep read-only attribute)   /Y (suppress prompt)
/L Copy symbolic links   /Z (overwrite)
/LD (create link)

See also: ATTRIB, MOVE, and REN.

File Selection

Supports attribute switches, extended wildcards, ranges, multiple file names, delayed variable expansion, and include lists. 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 for details.

Internet

Can be used with FTP / FTPS / TFTP / HTTP / HTTPS Servers.

Usage

If you don't specify any arguments, COPY will display its command dialog.

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.

If you specify the /C, /CF, /R, /U, or /UF options, COPY will append a **!** to the copy specifier if the target exists and is being overwritten. For example:

```
[d:\] copy file1 file2
file1 =>! file2
```

To emulate an approach used by some implementations of CMD, see the COPYCMD topic.

COPY sets three internal variables:

%_copy_dirs   The number of directories created
%_copy_files   The number of files copied
%_copy_errors   The number of errors

- **Copying Files**

You can copy several files at once by using wildcards:

```
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, 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 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:\MEM\MEMO2* and *C:\MEM\MEMO3* to *D:\DATA\MEMO1*, and leave the result in *C:\MEM\MEMO1*:

```
[c:\mem] copy d:\data\memo1+memo2+memo3
```

You cannot append files to a device (such as a printer); if you try to do so, COPY will ignore the + signs and copy the files individually. If you attempt to append several *source* files to a *destination* directory or disk, COPY will append the files and place the copy in the new location with the same name as the first *source* file.

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 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 switch. File descriptions are not copied when copying files to an Internet URL.

COPY supports the special syntax

```
copy con: ftp:...
```

to directly copy text from the console to an ftp location.

Wildcard characters such as ∗ and ? will be treated as wildcards in FTP URLs, but will be treated as normal characters in HTTP URLs.

Note: The /G option (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 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 and IFTP.

- **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 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 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:..   Forces COPY to process *source* files with the attributes you specify after the :, or to process all *source* files regardless of attributes, if /A: is used by itself.

/H   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   Retains the read-only attribute from each *source* file when creating the *destination* file. See /K below for a special note if you are running under Novell NetWare.

/Z   Forces COPY to overwrite an existing *destination* file regardless of its attributes.

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-style "*extension*" matching (assumes only one extension, at the end of the word) and the advanced TCC-RT string matching (allowing things like **.*.abc*) when an asterisk is encountered in the *destination* of a COPY command.

COPY supports regular expression back references in the target name. If you are using back references, you must also use a regular expression in the source name. The syntax is:

```
copy ::filename ::target
```

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\NoFileFolderConn
ection=0
```

You can override the default HTTP proxy server, proxy user, and proxy password (set in TCMD.INI) with the /Proxy... options.

/Proxy=*server*
/ProxyUser=*username*
/ProxyPwd=*password*

Options

The /A (ASCII copy) and /B (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 or /B, 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 and /P, COPY will generally take a "conservative" approach and give priority to the option which generates more prompts or more information.

/=   Display the COPY command dialog to help you set the filename and command line options. The /= option can be anywhere on the line; additional options will set the appropriate fields in the command dialog.

/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 option.

/A:...   Select only those files that have the specified attribute(s) set. See Attribute Switches 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 switch, above. Do not use **/A:** with **@file** lists. See @file lists for details. Hidden or system files selected by this option overwrite hidden or system files.

You can specify **/A:=** to display a dialog to help you set individual attributes.

/B   If you use **/B** with a *source* filename, the entire file is copied; Ctrl-Z characters, if any, in the file are considered ordinary data to be copied. Using **/B** with a *destination* filename prevents addition of a Ctrl-Z to the end of the *destination* file. **/B** is the default unless source files are appended to the target file, or the target is a device, e.g., **NUL**.

This option applies to the filename immediately preceding it, and to all subsequent filenames until the file name preceding the next /A or /B option.

/BAK If the target file exists, COPY will save it with a ".bak" extension before overwriting it. COPY will **not** create multiple versions of the .bak file; if you already have a *file.ext.bak*, it will be overwritten.

/C   Copy files only if the *destination* file exists and is older than the *source* (see also /U). This option is useful for updating the files in one directory from those in another without copying any files not already in the target directory. Before using **/C** in a network environment, be sure to read the note under /U. Do not use **/C** with *@file* lists. See @file lists for details.

/CDA Copy the attributes from each of the source subdirectories to the target subdirectories.

/CF   Copy files only if the *destination* file exists and is more than 2 seconds older than the *source* (see also /C and /UF). Do not use **/CF** with *@file* lists. See @file lists for details.

/CRC Create a file that contains a CRC + file name for every file copied.

type - The type of CRC to create. Possible types are:
>   MD5
>   CRC32
>   SHA1
>   SHA256
>   SHA384
>   SHA512

filename - The file that contains the CRC and file names (one per line).

/D   Force copy of an encrypted file even when the target will be decrypted (for CMD compatibility).

/DD   Remove any empty directories created with the /S option.

/DS   Change the date timestamp on the target file(s) to the specified date.

/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 / HTTP to ensure the copy is proceeding. When /V is also used, reports percentage verified.

/GZ   When copying to an HTTP / HTTPS target, COPY will compress the file using gzip before uploading it.

/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 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. **/J** will not work with HTTP or FTP files.

/K   (Keep read-only attribute) To maintain compatibility with CMD, 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   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 switch, or use ATTRIB. Do not use /M with *@file* lists. See @file lists 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.

    A **/N** with one or more 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)
        n   Don't copy/update the file descriptions
        r   A COPY /W will delete to the recycle bin (unless the file matches the RECYCLEEXCLUDE environment variable).
        s   Don't display the summary.
        t   Don't update the CD / CDD extended directory search database *(JPSTREE.IDX).*
        z   Skip system directories (when used with /S)

/O   Only copy the source file if the target file doesn't exist.

/O:...   Sort the files before processing.

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.

    -   Reverse the sort order for the next sort key

    a   Sort names and extensions in standard ASCII order, instead of numerically when numeric substrings are included in the name or extension.

    c   Sort by compression ratio

    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 description

    o   Sort by owner

    r   Reverse the sort order for all options

    s   Sort by size

    t   Same as **d**

    u   Unsorted

    z   Same as **s**

The /O:... option saves all of the matching filenames and then performs the copy. This avoids the potential problem of copying files more than once.

/P   Ask the user to confirm each *source* file. Your options at the prompt are explained in detail under Page and File Prompts. Note: the Copy Prompt on Overwrite configuration option can be used to force prompting at the command line only. See also: the /Q option below.

/Proxy=*server*

/ProxyUser=*username*

/ProxyPwd=*password*

/Q   Don't display filenames, percentage copied, total number of files copied, etc. 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 the user before overwriting an existing file. Your options at the prompt are explained in detail under Page and File Prompts. See also: the Copy Prompt on Overwrite configuration option. (For compatibility with CMD, a /Y option on the command line is changed to /R.)

/RCT   Request the transfer channel compress the data during the copy operation. In Windows 10+, this option is supported for files residing on SMB shares where the SMB protocol version is v3.1.1.1 or greater.

/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 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 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.

If you specify a **+** followed by a number after the /S, COPY will not copy any files until it gets to that depth in the subdirectory tree. For example, if you have a directory tree \a\b\c\d\e, /S+2 will not copy anything in \a or \a\b.

/SX   Copy the subdirectory tree to a single target directory (implies /S). For example, to copy all of the .EXE files in "c:\files" and all of its subdirectories to the directory "d:\exefiles":

```
copy /sx c:\files\*.exe d:\exefiles\
```

/T   Turns off the display of filenames, like /Q, but does display the total number of files copied.

/TS   Change the time timestamp on the target file(s) to the specified time.

/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). 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 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.

/UF   Copy each *source* file only if it is more than 2 seconds newer than a matching *destination* file or if a matching *destination* file does not exist (see also /CF and /U). Do not use **/UF** with @file lists. See @file lists for details.

/Vn   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. /V will not work for FTP, TFTP, or HTTP copies. If *n* is set, it specifies the number of retries (0-*n*) if the verification fails. If *n* is specified and all of the retries fail, the target file will be deleted.

/W   Delete files in the target directory that don't exist in the source directory. (Use this instead of SYNC when you only want to synchronize "one way".)

/WAIT=*n* Pause for *n* milliseconds between each block copied from the source to the target file. This is useful for users with slow networks and very large file copies; it prevents COPY from monopolizing all of the network I/O.

/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. **/X** should not be used with multiple targets, because the archive attribute will be cleared after the first copy.

/Y   If you have the COPY Prompt on Overwrite option set, you can suppress the prompt with /Y.

/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](#) or [/A:](#) is used to select it.

4.2.29   COPYDIR

Purpose: Copy a directory tree

Format: COPYDIR *source destination*

> **source**   The source directory tree
> **destination**   The target directory tree

Usage:

Both *source* and *destination* must be directory names. If *destination* does not exist, COPYDIR will create *destination* and copy *source* to *destination*. If *destination* already exists, COPYDIR will append the last subdirectory name in *source* to *destination,* create the new subdirectory, and copy *source* to *destination*. (This allows you to rename the target directory.)

Example:

To copy *d:\test\mydir* to *x:\mydir:*

```
copydir d:\test\mydir x:\
```

To copy *d:\test\mydir* to *x:\myolddir:*

```
copydir d:\test\mydir x:\myolddir
```

4.2.30   DATE

Purpose: Display and optionally change the system date

Format: DATE [/Fn /T /U *"format"*] [*mm -dd -yy*]

> **mm**   The month (1 - 12)
> **dd**   The day (1 - 31)
> **yy**   The year (80 - 99 or a 4- digit year)

> [**"..."**](#)   Date display format

[/F(ormat)](#)   [/U (UTC date)](#)
[/T (Display only)](#)

See also: [TIME](#).

Usage:

If you simply type DATE without any parameters, you will see the current system date and time, and be prompted for a new date. Press **Enter** if you don't wish to change the date. If you type a new date, it will become the current system date, which is included in the directory entry for each file as it is created or altered:

```
date
Wed 16/06/2021  10:40:36a
New date (mm-dd-[yy]yy):
```

You can also enter a new system date by typing the DATE command plus the new date on the command line:

```
date 30/06/2021
```

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**.

The day of week and month are translated into your local language (English, French, German, Italian, Russian, and Spanish).

Options:

"..." Custom date / time format to use when displaying the current date. The formatting characters are the same as used by the @DATEFMT function.

/F Date format to use. The formats are:

> 0 : Tue Jan 1, 2019
> 1 : 1/01/19
> 2 : Tue 1/01/2019
> 4 : 2019-01-01

/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.

/U Display or enter the UTC date.

4.2.31 DATEMONITOR

Purpose: Monitor the current date and time

Format: DATEMONITOR [/C [*yyyy-mm-dd hh:mm*]]
DATEMONITOR *yyyy-m-dd hh:mm n command*

> **n** Number of repetitions (or **FOREVER**)
> **yyyy-mm-dd** The date to match
> **hh:mm** The time to match

command	Command to execute when the specified date and time matches the current date and time

/C(lear)

Usage:

DATEMONITOR monitors the current date and time, and executes the specified command when the current date and time match the saved date and time. You can use a * in the date fields if you want to run a command at a specific time every day. For example:

```
datemonitor *-*-* 23:59 forever echo It's almost midnight!
```

If you want to run a command on the first day of every month:

```
datemonitor *-*-1 00:01 forever echo It's the beginning of a new month!
```

If you don't enter any arguments, if DATEMONITOR is active it will display the repeat count and the command.

DATEMONITOR sets two environment variables when the date and time match and the trigger is set:

_datemonitor	The current date in yyyy-mm-dd format
_timemonitor	The current time in hh:mm (24-hour) format

The command line will be parsed and expanded before DATEMONITOR 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 the last argument on the line is a single **(,** it is interpreted as the beginning of a command group. DATEMONITOR will append the following lines (in a batch file) or prompt you for more input (at the command line) until it gets a closing **).**

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-RT thread is also performing I/O. You may need to use START or DETACH in ***command*** to avoid conflicts.

Options:

/C	Remove date monitors. You can optionally specify a specific date monitor to remove by entering the date and time (which may include wildcards) for that monitor.

4.2.32 DEBUGMONITOR

Purpose:	Monitor the OutputDebugString API

Format:	DEBUGMONITOR [/C] DEBUGMONITOR *n command*

n	Number of repetitions (or **FOREVER**)
command	Command to execute when condition is triggered

/C(lear)

Usage:

DEBUGMONITOR looks for any process calling the Windows OutputDebugString API.

DEBUGMONITOR will set the environment variable _OUTPUTDEBUGSTRING to the value specified in the OutputDebugString call.

The command line will be parsed and expanded before DEBUGMONITOR 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 the last argument on the line is a single **(,** it is interpreted as the beginning of a command group. DEBUGMONITOR will append the following lines (in a batch file) or prompt you for more input (at the command line) until it gets a closing **).**

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-RT thread is also performing I/O. You may need to use START or DETACH in **command** to avoid conflicts.

Options:

/C   Remove the OutputDebugString monitor.

4.2.33 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.

4.2.34 DEDUPE

Purpose: Search for and optionally delete or symlink duplicated files.

Format: DEDUPE [*ranges*] [/A:[[-|+]rhsadecijopt /D /L /N[defjnstz] /P /Q /R /S[[+]
n] /SHA1 /SHA256 /SHA384 /SHA512 /T /V /W[n]] *filename directory* [*directory*...]

filename The filename to search for (* for everything)
directory The directories (and optionally subdirectories) to search

/A:... (attributes)	/R (recycle)
/D (delete)	/S[n] (subdirectories)
/L (symlinks)	/SHAx (Hash type)
/N (disable)	/V (verbose)
/P (prompt)	/Wn (wipe)
/Q (quiet)	

File Selection

Supports attribute switches, extended wildcards and, ranges. Use wildcards with caution on LFN volumes; see LFN File Searches for details.

DEDUPE searches one or more directories, assigns a hash value to files, and then compares the hash value to all the other files. On slow systems (and particularly in x86 Windows) this can take a while, so you should try to reduce the amount of searching & hashing by using ranges, and not try to dedupe an entire disk at one time.

DEDUPE assumes that the first file it finds for each hash is the original file.

Example:

Remove all the duplicate files from the E: drive:

```
dedupe /d /s /SHA256 * e:\
```

Options:

/A: Dedupe those files that have the specified attribute(s) set. See Attribute Switches for information on the attributes which can follow **/A:**. Do not use /A: with @file lists. See @file lists for details.

          You can specify **/A:=** to display a dialog to help you set individual attributes.

/D   Delete duplicate files

/L   Convert duplicate files to symlinks of the first file. Note that to create symlinks, you must be in an elevated session.

/N   Change default options. This can be any combination of the following:

          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
          z   Skip system directories (when used with /S)

/P   Prompt before deleting or symlink'ing files.

/Q   Quiet (don't display directories or files as they are processed)

/R   Delete to the recycle bin

/S*n*   Search subdirectories

/SHA*x*   Hash algorithm to use. The default is SHA256; you can optionally use SHA1 or SHA512. SHA1 is slightly faster but potentially insecure.

/V   Verbose output

/W*n*   Wipe deleted files

4.2.35 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.

If the last argument on the line is a single **(,** it is interpreted as the beginning of a command group. DEFER will append the following lines (in a batch file) or prompt you for more input (at the command line) until it gets a closing **).**

4.2.36 DEL / ERASE

Purpose: Erase one file, a group of files, or entire subdirectories

Format: DEL [*ranges*] [/A:[[-|+]rhsadecijopt /E /F /I"*text*" /K /L /N[defjnstz] /O:[-] adegnrstu /P /Q /R /S[[+]n] /T /W[n] /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)	/P(rompt)
/B (Delete after reboot)	/Q(uiet)
/E (No error messages)	/R(ecycle bin)
/F(orce delete)	/S(ubdirectories)
/I (match descriptions)	/T(otal)
/K (no Recycle Bin)	/W(ipe)
/L (delete symlinks)	/X (remove empty subdirectories)
/N (Disable)	/Y(es to all prompts)
/O:... (Order)	/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, behaves the same way but does not ask for confirmation if you use /Q to delete files quietly. If you want TCC-RT to follow CMD'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 option, which overwrites the file before deleting it. Use this option with caution! Once a file is obliterated, it is impossible to recover. Remember: /W 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 % ? internal variable, and use it with conditional commands (**&&** 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 for additional details.

If you are deleting a stream, DEL will check for a symlink and delete the stream from the linked file. (Windows does not support deleting a symlink'd stream.)

DEL sets three internal variables:

%_del_dirs	The number of directories deleted
%_del_files	The number of files deleted
%_del_errors	The number of errors

- **Recycle Bin**

When you delete files with DEL, TCC-RT does not move the deleted files to the Windows Recycle Bin by default. You can change this default with the Delete to Recycle Bin 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 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 option:

```
del /k letters.txt
```

You can also exclude files from the Recycle bin, even if Delete to Recycle Bin is enabled, or if the command use the /R option, with the RecycleExclude environment variable.

If you are deleting to the recycle bin, the DEL result will say "xx files sent to the recycle bin" instead of "xx files deleted".

- **FTP Usage**

If you have appropriate permissions, you can delete files on FTP servers. For example:

```
del ftp://ftp.abc.com/index
```

You can also use the IFTP 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 for additional details.

Options

/A: Delete only those files that have the specified attribute(s) set. See Attribute Switches for information on the attributes which can follow **/A:**. Do not use /A: with @file lists. See @file lists for details.

     You can specify **/A:=** to display a dialog to help you set individual attributes.

/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 (see below): it deletes read-only, hidden, and system files as well as normal files.. It is included for compatibility with CMD.

/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"[]"**.

/K   Physically delete files instead of sending them to the Windows Recycle Bin.

/L   Delete symlinks instead of their contents.

/N   Do everything except actually delete the file(s). This is useful for testing the result of a DEL.

     A **/N** with one or more 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)
       n   Don't update the file descriptions
       s   Don't display the summary
       z   Skip system directories (when used with /S)

/O:...   Sort the files before processing.

     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, instead of numerically when numeric substrings are included in the name or extension.
d	Sort by date and time (oldest first); also see **/T:acw**
e	Sort by extension
g	Group subdirectories first, then files
r	Reverse the sort order for all options
s	Sort by size
t	Same as **d**
u	Unsorted

/P   Prompt the user to confirm each erasure. Your options at the prompt are explained in detail under Page and File Prompts.

/Q   Don't display filenames as they are deleted, or the number of files deleted or bytes freed. If Prompt on Wildcard Deletes is disabled then **/Q** also disables the normal confirmation prompt when performing wildcard deletions (*e.g.* DEL *), for compatibility with CMD. Use caution if you disable Prompt on Wildcard Deletes, as this will allow DEL /Q to delete an entire directory without prompting for confirmation. See also /T.

/R   Delete files to the Windows Recycle Bin.

/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 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.

If you specify a **+** followed by a number after the /S, DEL will not delete any files until it gets to that depth in the subdirectory tree. For example, if you have a directory tree \a\b\c\d\e, /S+2 will not delete anything in \a or \a\b.

/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.

/W[n]   Overwrite the file contents using the DoD 5220.22-M (E) standard for secure deletion. (This overwrites every byte in the file with different values). You can optionally specify the number of passes (1-999); the default is 3. 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 will override a /R.

/X   Removes empty subdirectories (only useful when used with /S). If DEL deletes one or more directories, they will be removed automatically from the extended directory search database. DEL will display the directories being removed (with a trailing \).

/Y   The reverse of /P . It assumes a Y response to everything, including deleting an entire subdirectory tree. **TCC-RT** 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](#) works by hiding files, /Z will override an [EXCEPT](#) command. However, files specified in a [file exclusion range](#) 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\
```

4.2.37　DELAY

Purpose:　Pause for a specified length of time

Format:　DELAY [/B /F /M *time*]
　　　　　　DELAY UNTIL [*yyyy-mm-dd*] *hh:mm*[:*ss*]

　　　　　　time　The number of seconds or milliseconds to delay.

　　　　　　[/B(reak enabled)](#)　　　　　[/M(illiseconds)](#)
　　　　　　[/F(lush keyboard)](#)

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 maximum **time** value is limited to about 585 million years in Windows. If you don't enter a **time**, the default is 1 second.

You can optionally wait until the specified date and time. If no date is specified, DELAY defaults to today.

TCC-RT 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.
/F	Flush the keyboard buffer when DELAY ends.
/M	Count by milliseconds instead of seconds. Normally only used for delays of less than 1 second.

4.2.38 DESCRIBE

Purpose: Create, modify, or delete file and subdirectory descriptions

Format: Creating or modifying descriptions
DESCRIBE [*ranges*... /I"*text*"] [/A:atrlst /O:[-]adegnrstu /C*n* /R /W] [@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 for details).
"description"	The description to attach to the file.

/A: (Attribute select)	/O:... (Order)
/D(escription follows)	/R(emove)
/C*n* (convert description format)	/U(pdate) descriptions file
/I (match description)	/W (description dialog)

See also: @DESCRIPT, DIR, and SELECT

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.

Usage:

DESCRIBE adds descriptions to files and subdirectories. (Volume root directories cannot have descriptions.) The descriptions are displayed by DIR in single-column mode and by SELECT, and can be retrieved using the @DESCRIPT 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 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-RT will treat the string as a quoted file name, not as description text as you intended. To resolve this problem use the **/D** switch immediately prior to the quoted description (with no intervening spaces). For example, if the current directory contains the files *DATA.TST* and *"Test File"*, the first of these commands will work as intended, but the second will not (in the second example the string "test file" will be treated as a second file name, when it is intended to be description text):

```
describe data.tst /D"test file"    correct command
describe data.tst "test file"      incorrect command
```

On LFN drives you will not see file descriptions in a normal DIR display, because DIR must leave space for the long filenames. To view the descriptions, use DIR /Z to display the directory in FAT format. See DIR for more details.

Each description can be up to 511 characters long. You can change this limit with the Maximum Length 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 output). DESCRIBE can edit descriptions longer than Maximum Length (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 configuration option), or in each directory in a hidden file called *DESCRIPT.ION*. Use the ATTRIB command to remove the hidden attribute from this file if you need to copy or delete it. *DESCRIPT.ION* is always created as a hidden file, but will not be rehidden by TCC-RT if you remove the hidden attribute.

The description file is modified appropriately whenever you perform an internal command which affects it (such as COPY, MOVE, DEL, or RENAME), but not if you use an external program (such as XCOPY or Explorer). You can disable description processing with SETDOS /D.

When you COPY, MOVE or REN files between two directories, both of which have descriptions, and you use switches which enable processing of hidden files (or you have removed the hidden attribute from DESCRIPT.ION), you must use caution to avoid overwriting existing file descriptions in the ***destination*** directory with the DESCRIPT.ION file from the ***source*** directory. See the notes under the **Advanced Features** sections of COPY and MOVE for additional details.

If you disable descriptions with the SETDOS /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 for information on the attributes which can follow **/A:**. Do not use **/A:** with @file lists. See @file lists for details.

You can specify **/A:=** to display a dialog to help you set individual attributes.

/C*n dir* Convert descriptions between DESCRIPT.ION and the NTFS file summary formats. The argument following /C*n* is the start directory; DESCRIBE will convert the descriptions in that directory and all of its subdirectories.

    /C0   convert descriptions from NTFS to DESCRIPT.ION
    /C1   convert descriptions from DESCRIPT.ION to NTFS

/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](#) 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](#) for details.

/O:... Sort the files before processing. 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, instead of numerically when numeric substrings are included in the name or extension.
    d   Sort by date and time (oldest first); also see **/T:acw**
    e   Sort by extension
    g   Group subdirectories first, then files
    r   Reverse the sort order for all options
    s   Sort by size
    t   Same as **d**
    u   Unsorted

/R Remove the old description after converting (only valid when used with /C*n*).

/U Update the *DESCRIPT.ION* file, 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.

/W Description editor dialog for creating / editing / deleting descriptions.

4.2.39 DESKTOP

Purpose: Create or switch desktops

Format: DESKTOP [/C [/N] *newdesktopname* [,*startapp*]]
                      DESKTOP *desktopname*

        startapp   The program to launch in the new desktop.

/C(reate)   /N(o activate)

Usage:

DESKTOP will create a new Windows desktop, or switch to an existing desktop.

You can specify the program DESKTOP should launch in the new desktop. The default is USERINIT.EXE (which will launch Windows Explorer).

If you don't specify any arguments, DESKTOP displays all of the existing desktops.

NOTE: DESKTOP does not support the Virtual Desktops introduced in Windows 10.

Options:

/C   Create a new desktop.

/N   Don't switch to the new desktop.

4.2.40 DETACH

Purpose: Start a console (character-mode) application in detached mode

Format: DETACH [/Q] *command*

/Q(uiet)

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 and TASKEND.

Usage:

When you start a program with DETACH, that program cannot use the keyboard, mouse, or video display. It is "detached" from the normal means of user input and output. However, you can redirect the program's standard I/O to other devices if necessary, using redirection symbols. 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-RT will detach a copy of TCC-RT to execute the command.

Once the program has started, TCC-RT returns to the next command immediately. It does not wait for a detached program to finish.

The Process ID of the detached program is returned in the _DETACHPID variable.

You can use the TASKEND command to stop a detached program which does not terminate on its own.

Examples:

The following command will detach a copy of TCC-RT 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"
```

If you prefer, you can use the Linux syntax of putting a trailing **&** on the command line instead of specifying DETACH. (TCC-RT will convert it to DETACH before executing the command.)

```
xyz.btm &
```

Options:

    /Q   Don't display the new process's ID.

4.2.41 DIFFER

    Purpose: Compare directories

    Format: DIFFER [*ranges*] [/A:[[-|+]rhsadecijopt /A /C /D /N[ejs] /S *source target*

                source - source directory
                target - target directory

/A:... (Attributes)	/N (disable)
/A (Added files)	/S[n] (Subdirectories)
/C (Changed files)	/SHAx (Hash type)
/D (Deleted files)	

Usage:

DIFFER will compare two directories and display files that have been added, changed, or deleted. If you don't specify the /A, /C, and/or /D options, DIFFER will prefix the line with a [*] for changed files, [+] for added files, and [-] for deleted files.

DIFFER will not by default search *target* to see if there are additional directories that are not in *source*. You can either reverse the source & target, or specify the /A:D option if you want to find new target subdirectories.

Examples:

Compare the directories C:\DEV and D:\DEV (and their subdirectories) and display the differences:

```
differ /S /SHA256 c:\dev d:\dev
```

Options

    /A: Compare only those files that have the specified attribute(s) set. See Attribute Switches for information on the attributes which can follow **/A:.**

    /A   Only look for files that have been added to the target directory.

/C   Only look for files that have been changed in the target directory.

/D   Only look for files that have been deleted from the target directory,

/N   A **/N** with one or more of the following arguments has an alternate meaning:

> **e**   Don't display errors
> **j**   Skip junctions (when used with /S)
> **s**   Don't display the summary

/S   Also compare subdirectories in *source* and *target*. If you specify a number after the **/S**, DIFFER will limit the subdirectory recursion to that depth. For example, if you have a directory tree "\a\b\c\d\e", **/S2** will only compare the "a", "b", and "c" directories.

/SHAx   Hash type to use for the file comparison. You can use /SHA1, /SHA256, or /SHA512. If you don't specify a hash type, then DIFFER will compare the file times (which is much faster, but can't determine that two files are identical if the date/times are different).

4.2.42 DIR

Purpose: Display information about files and subdirectories

Format: DIR [*ranges*] [*options*] [*file*...]

> ***ranges***   one or more ranges
> ***options***   one or more file selection or report format options
> ***file***   The file, directory, or list of files or directories to display.

/1	1 column output	/L	Lower case
/2	2 column output	/M	suppress footer
/4	4 column output	/N[desfhjlmvz]	New format or disable options
/:	show streams	/O	Order
/A	Attribute select	/P[*n*]	Pause
/B	Bare (name only)	/Q	show owner
/C	show Compression	/R	disable wrap
/CD:...	COLORDIR string	/Sn	show Subdirectories to depth *n*
/D	Disable color coding	/T	show aTtributes
/E	upper case	/T:	time type
/F	Full path	/U	show summary information
/G[:n]	allocated size	/V	Vertical sort
/H	Hide dots	/W	Wide
/HL	Show hard links	/X	show short names
/I"text"	description range	/Z	use FAT format
/J	Justify names	/\	Trailing \ for directory names
/K	suppress header		

See also: ATTRIB, DESCRIBE, PDIR, SELECT, and SETDOS.

File Selection

Supports extended wildcards, ranges, multiple file names, and include lists.

Internet: Can be used with HTTP and FTP servers.

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 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 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.

DIR sets three internal variables:

%_dir_dirs	The number of directories displayed
%_dir_files	The number of files displayed
%_dir_errors	The number of errors

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
- ▶ Default DIR Output Format
- ▶ Switching Formats
- ▶ Multiple Column Displays
- ▶ Color-Coded Directories
- ▶ Redirected Output
- ▶ Other Notes
- ▶ Options
- ▶ FTP usage

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. 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 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 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](#) 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\version28\*

2021-06-30   0:39         <DIR>    .
2021-06-30   0:39         <DIR>    ..
2021-06-25  11:30         <DIR>    tcmd
2021-06-25  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>        6-24-21  12:17
..            <DIR>        6-24-21  12:17
TEST          <DIR>        6-01-21  16:21
jpstree.idx    196967      6-28-21  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	**/2** or **/4 columns**	**/W (wide)**
Normal	date, time, size, LFN	2 - 4 columns, LFNs only	No. of columns based on longest LFN
/Z (FAT)	truncated LFN, size, date, time	2 - 4 columns, truncated LFN plus date, time, size	5+ columns, truncated LFNs only
/X (SFN)	date, time, size, SFN, LFN	2 - 4 columns, SFNs plus date, time, size	5+ columns, SFNs only
/X /Z	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, attributes, or matching range.

To choose the display colors, you must either use the SET command to create an environment variable called COLORDIR, or use the Directory Colors 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 option. You may find it useful to use the COLORDIR variable for experimenting, then to set permanent directory colors with the Directory Colors option.

The format for both the COLORDIR environment variable and the Directory Colors option is:

```
ext ... :ColorName; ...
```

where "ext" is either a file extension (which may include wildcards), 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

or a range (size, date, time, description, owner, and/or exclusion), or a file subsystem type:

EXETYPE_WIN32GUI	Windows x86 GUI app
EXETYPE_WIN32CUI	Windows x86 console app
EXETYPE_WIN64GUI	Windows x64 GUI app
EXETYPE_WIN64CUI	Windows x64 console app
EXETYPE_DOS	DOS (16-bit) app (obsolete)
EXETYPE_POSIX	POSIX app (obsolete)
EXETYPE_EFI	EFI app

and "ColorName" is any valid color name (see Colors and Color Names for information on color names). Specifying a subsystem type will significantly slow down the directory display, as TCC-RT has to read the header of each file to find a match.

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=exe:red; c asm:bright cyan; rdonly:green on white
```

To display 32-bit console apps in bright green and 64-bit console apps in bright red:

```
set colordir=EXETYPE_WIN32CUI:bri green;EXETYPE_WIN64CUI:bri red
```

Extended wildcards 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 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. For example:

```
dir ftp://ftp.microsoft.com/
```

You can also use the IFTP command to start an FTP session on a server, and then use a simplified syntax to specify the files and directories you want.

HTTP Usage

DIR has limited support for HTTP & HTTPS filenames. DIR will display the filename, size, and date/time (for last write only). Wildcards are not supported (this is an HTTP limitation, not **TCC-RT**).

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.

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).

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.

/   Display directory names with a trailing \.

/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. 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 for information on the attributes which can follow **/A:**.

You can specify **/A:=** to display a dialog to help you set individual attributes.

/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.**

        /B1 will display relative paths when used with /S. (Normally, /B shows the full pathname for the file.)

/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.

/CD: Define a custom directory colorization string to use instead of the COLORDIR environment variable, or the ColorDir option in TCMD.INI.

/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[:n]   Display the allocated disk space instead of the actual size of each file. You can optionally specify the disk cluster size to be used by /G. (DIR will normally query the system for the cluster size on the specified drive, but you can override with /G:n if you know that the returned info is incorrect, or if you want to find the size required if the specified files were moved to another device with a different cluster size.)

/H   Suppress the display of the "**.**" and "**..**" directories.

/HL   Show the hard links for files and directories. **/HL** can only be used in single-column mode.

/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
h	Don't display the header
j	Skip junctions (when used with /S)
l	Don't display the link name for symbolic links
m:n	Display a maximum of *n* directory entries
s	Don't display the summary.
v	Don't display the volume information*.*
z	Skip system directories (when used with /S)

/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, instead of numerically 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
t	Same as **d**
u	Unsorted
x	When combined with /S, sorts the results from all directories together and displays them in a single listing. (Unless you're also specifying /F, you probably won't get a comprehensible result.) Note that /O:x will turn off headers and footers.

/P[*n*]   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. The /P option has an optional argument *n* that specifies the number of seconds to wait for a keystroke before continuing.

/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.

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.

If you specify a **+** followed by a number after the /S, DIR will not display any filenames until it gets to that depth in the subdirectory tree. For example, if you have a directory tree \a\b\c\d\e, /S+2 will not display the contents of \a or \a\b.

/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	**C**	Compressed
E	Encrypted	**O**	Offline
N	Normal	**I**	Not content-indexed
T	Temporary	**L**	Junction or symbolic link
P	Sparse file		

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-RT 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 an 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](#), [/4](#) or [/W](#) 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-RT 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 **SFN**). /X also selects short filenames in the /2, /4, /B, /W , and /Z displays, and short file and path names in the /F display.

/Z   Display filenames on LFN drives in the old-style format, with the filename on the left and the description (when available) on the right. Long names will be truncated to 12 characters unless /X is also used. If the name is longer than 12 characters, it will be followed by a ➔ "right arrow" symbol to show that one or more characters have been truncated. If a description file exists, /Z defaults to using the name of the . and .. directories as description for those entries

4.2.43 DIRENV

Purpose: Configures the environment on a per-directory basis.

Format: DIRENV [ON | OFF]

Usage:

When DIRENV is on, TCC-RT will look for a file called ".envtc" when it changes directories. The format of .envtc is:

    var1=value1
    var2=value2
    ...

When TCC-RT enters a directory, it will look in .envtc for environment variables to set while in that directory. When TCC leaves a directory, it will unset the variables in .envtc.

You can force DIRENV to always enabled with the DirEnv .INI directive.

4.2.44 DIRS

Purpose: Display the current directory stack

Format: DIRS [+n -n /M /P /Q] [*name*]

            +n / -n   Rotate the directory stack up or down **n** entries
            name   Display only directories which match *name*

            /M (number lines)
            /P(ause)
            /Q(uiet)

See also: PUSHD, POPD, and @DIRSTACK.

Usage:

The PUSHD command adds the current default drive and directory to the directory stack, a list maintained by TCC-RT. The POPD command removes the top entry of the directory stack and makes

that drive and directory the new default. The DIRS command displays the contents of the directory stack, with the most recent entries last (*i.e.*, the next POPD will retrieve the last entry that DIRS displays).

The name to match can include wildcards.

The directory stack holds 16K characters, enough for 400+ typical drive and directory entries.

Examples:

To change directories and then display the directory stack:

```
[c:\] pushd c:\database
[c:\database] pushd d:\wordp\memos
[d:\wordp\memos] dirs
c:\
c:\database
```

You can optionally display only those directories in the stack which match a name. For example:

```
DIRS c:            (only display directories on the C: drive)
DIRS \\server\share   (only display directories on this UNC share name)
```

Options

    /M   Number the lines when displaying the DIRS 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.

    /Q   Don't display the directory stack (only useful when combined with +n or -n).

4.2.45 DISKMONITOR

    Purpose: Monitor free disk space

    Format: DISKMONITOR [/C [*disk*]]
                      DISKMONITOR disk size command

             disk   Disk drive to monitor
             size   Minimum free disk space
             command   Command to execute when condition is triggered

             /C(lear)

Usage:

If the free disk space for the drive drops below the specified size, DISKMONITOR will execute the specified command. For example, to send an email when the C: drive has less than 2Gb free:

```
DISKMONITOR C: 2Gb sendmail bob@bob.com "Disk Status" Drive C: is full!
```

The drive can also be a sharename. The size format is the same as that used for size ranges (i.e., either a number or a number with an appended k, K, m, M, g, G, t, or T).

The command line will be parsed and expanded before DISKMONITOR 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 the last argument on the line is a single **(,** it is interpreted as the beginning of a command group. DISKMONITOR will append the following lines (in a batch file) or prompt you for more input (at the command line) until it gets a closing **).**

If you don't enter any arguments, DISKMONITOR will display the disk drives it is currently monitoring.

DISKMONITOR will poll the drives it is monitoring once every 10 seconds.

Options:

/C   If *disk* is specified, remove the monitor for that disk drive. Otherwise, remove all disk monitors.

4.2.46 DNS

Purpose: Display the DNS records for the specified DNS server and host domain.

Format: DNS [/Nh] *server hostname*

         server - The address of the DNS server
         hostname - The host domain to query

Usage:

For example:

     DNS 1.1.1.1 jpsoft.com

4.2.47 DO

Purpose: Create loops in batch files

Format: DO *loop_control*
         commands
     [ITERATE]
         commands
     [LEAVE [n]]
         commands
     ENDDO

Loop_control formats

         DO count
         DO FOREVER
         DO varname = start TO end [BY step] [(command)]
         DO WHILE condition [(command)]
         DO UNTIL condition [(command)]

DO UNTIL DATETIME date time [(command)]
DO FOR n [SECONDS | MINUTES | HOURS] [(command)]
DO varname IN [range...] /D"directory" [/I:"text" /S[[+]n] /A:[[-|+]rhsadecijopt /O:[-]
adegnrstu fileset [(command)]
DO varname IN [/T"delimiters"] /L stringset [(command)]
DO varname IN /C stringset [(command)]
DO varname IN /L stringset [(command)]
DO varname in /P command ... [(command)]
DO varname IN /Q stringset [(command)]
DO varname IN @file [(command)]

count	Integer in the range [0, 2147483647], 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 [-2147483647, 2147483647] or internal variables or variable functions that evaluate to such values, controlling the number of times the loop is executed.
condition	A conditional expression to determine whether or not the loop should be executed
fileset	A filename or list of filenames, possibly using wildcards
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	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 8601 format
time	The loop termination time in 24-h *hh:mm:ss* format
/A:	Attribute select
/C	Loop through each character in expression
/D"directory"	Start directory
/I"text"	(Match description) Description range.
/L(iteral)	Members of *set* are strings, not filenames
/O:... (Order)	Sort order
/P	Parse the output of the command, saving the next output line in *varname* each time the loop is executed.
/Q	Like /L, but treats double quoted arguments (with embedded whitespace) as a single argument.
/S	Perform the loop in the current directory and all its subdirectories

Supports extended wildcards, ranges, and include lists for the *set*. Use wildcards with caution on LFN volumes; see LFN File Searches for details.

Usage

DO can be used in batch files, aliases, or at the command prompt. To use them in aliases or at the prompt, you need to define the DO on a single line, and enclose the body of the DO loop in a command group following the DO expression. (There is no ENDDO statement in a single-line DO). For example:

```
do count=1 to 10 by 1 (echo count=%count)
```

If the last argument on the line is a single **(**, it is interpreted as the beginning of a command group. DO will append the following lines (in a batch file) or prompt you for more input (at the command line) until it gets a closing **)**.

When you use DO on an LFN drive, you must quote any file names which contain white space or special characters. The same restriction may apply to names returned in the DO variable, if you pass them to TCC-RT internal commands, or other commands which require quoting filenames with white space. DO does not quote returned names automatically, even if you included quotes in the original argument.

DO sets four internal variables:

%_do_dirs	The number of directories traversed (with /S) for the current DO loop. (I.e., nested DO's each have their own _do_dirs, _do_files, _do_errors, and _do_loop.)
%_do_files	The number of directory entries (files or subdirectories) processed for the current DO loop.
%_do_errors	The number of errors for the current DO loop.
%_do_loop	The number of times the current DO loop has been executed.

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.

- DO FOREVER creates an endless loop. You must use LEAVE or GOTO to exit such a loop.

- DO *varname = start* TO *end* [BY *step*] is similar to a "for loop" in programming languages like BASIC. DO creates an environment variable, *varname*, and sets it equal to the value *start*. If *varname* already exists in the environment, it will be overwritten. DO then begins the loop process by comparing the value of *varname* with the value of *end*. If *step* is positive or not specified, and *varname* is less than or equal to *end*, DO executes the batch file lines up to the ENDDO. Next, DO adds to the value of *varname* either the value of *step* if BY *step* is specified, or **1**, and repeats the compare and execute process until *varname* is greater than *end*. This example displays the even numbers from 2 through 20:

```
do i = 2 to 20 by 2
 echo %i
enddo
```

DO can also count down, rather than up. If *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 **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](#) 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](#) 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)](#). 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-RT has no control.

  To use date, time, size, description, or file exclusion [ranges](#) for the *set* place them just before the filename(s), for example:

  ```
  do x in /[d9-1-2018,9-31-2018] *.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-RT, such as redirection and piping symbols, within the file (use [SETDOS](#) /X as needed).

  To execute the loop once for each line of text in the clipboard, use **CLIP:** (or CLIP0: - CLIP9:) as the file name (*e.g.* DO X IN @CLIP:). **CLIP:** will not return any data unless the clipboard contains text. See [Redirection](#) 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 or IFF command (group):

```
do while "%var" != "%val1"
 ...
 if "%var" == "%val2" leave
enddo
```

LEAVE accepts an optional numeric argument (>=1) which specifies the DO nesting level you want to leave. For example, "LEAVE 2" will exit two nested DO loops. You can optionally pass a variable as the LEAVE argument.

Usage Notes

Numeric input may be entered in either decimal format (a sequence of 0-9 digits) 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.

You can exit from all DO / ENDDO loops in a batch file by using GOTO to a line past the corresponding ENDDO. However, be sure to read the cautionary notes about GOTO and DO under the GOTO command before using GOTO in any other way inside any DO loop.

You cannot use RETURN to return from a GOSUB while inside a DO loop.

Note: Do not confuse the DO command with the unrelated optional *do* keyword of the FOR command.

Options:

/A: Select the files in a DO x IN ... by their specified attribute(s). See Attribute Switches for information on the attributes which can follow **/A:**.

You can specify **/A:=** to display a dialog to help you set individual attributes.

/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). /D supports Windows shell folder names; see CDD for details.

/I"*text"*** Select files in a DO x IN ... by matching ***text*** in their descriptions. See Description Ranges for details.

/L   The parameters following DO x 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. /L will not treat double quotes as delimiters; use /Q if you want to pass arguments with embedded white space.

/N   Disable options:

d   Skip hidden directories (when used with /S)

> **j** Skip junctions (when used with /S)

/O:... Sort the files before processing.

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, instead of numerically when numeric substrings are included in the name or extension.
> **d** Sort by date and time (oldest first); also see **/T:acw**
> **e** Sort by extension
> **g** Group subdirectories first, then files
> **r** Reverse the sort order for all options
> **s** Sort by size
> **t** Same as **d**
> **u** Unsorted

The /O:... option saves all of the matching filenames and then performs the requested operation. This avoids the potential problem of processing files more than once.

/P For each loop, assign the next output line from **command** to the DO variable.

/Q The parameters following DO x IN /Q 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. Unlike /L, /Q will treat double quoted arguments with embedded whitespace as a single argument.

/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.

If you specify a **+** followed by a number after the /S, DO will not execute **command** until it gets to that depth in the subdirectory tree. For example, if you have a directory tree \a\b\c\d\e, /S+2 will not execute **command** in \a or \a\b.

/T"text" Specify the delimiters to be used when parsing a string set.

4.2.48 DRAWBOX

Purpose: Draw a box on the screen

Format: DRAWBOX *ulrow ulcol lrrow lrcol style* [BRIght] *fg* ON [BRIght] *bg* [FILL [BRIght] *bgfill*] [ZOOm] [SHAdow]

> **ulrow** Row for upper left corner
> **ulcol** Column for upper left corner
> **lrrow** 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 and DRAWVLINE.

Usage:

DRAWBOX is useful for creating attractive screen displays in batch files.

See Colors and Color Names 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 (a sequence of 0-9 digits) 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-RT window. The maximum *column* value is determined by the current virtual screen width.

If *ulrow* is set to 999, *lrrow* is assumed to be the desired height, and the box will be centered vertically. If *ulcol* is set to 999, *lrcol* 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.

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
```

4.2.49 DRAWHLINE

Purpose: Draw a horizontal line on the screen

Format: DRAWHLINE *row column len style* [BRIght] *fg* ON [BRIght] *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 and DRAWVLINE.

Usage:

DRAWHLINE is useful for creating attractive screen displays in batch files. It detects other lines and boxes on the display, and creates the appropriate connector characters when possible (not all types of lines can be connected with the available characters).

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 (a sequence of 0-9 digits) 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-RT window. The maximum *column* value is determined by the current virtual screen width.

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 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.

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
```

4.2.50 DRAWVLINE

Purpose: Draw a vertical line on the screen

Format: DRAWVLINE *row column len style* [BRIght] *fg* ON [BRIght] *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 and DRAWHLINE.

Usage:

DRAWVLINE is useful for creating attractive screen displays in batch files. It detects other lines and boxes on the display, and creates the appropriate connector characters when possible (not all types of lines can be connected with the available characters).

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 (a sequence of 0-9 digits) 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-RT window. The maximum *column* value is determined by the current virtual screen width.

See Colors and Color Names 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.

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
```

4.2.51 ECHO

Purpose: Enable or disable batch file or command line echoing, display the echoing status on stdout, or display a message on stdout

Format: ECHO [ON | OFF | *message*]

message Text to display.

See also the commands ECHOS, ECHOSERR, ECHOERR, ECHOX, ECHOXERR, SCREEN, SCRPUT, TEXT and VSCRPUT, and the internal variable _ECHO.

Usage:

The ECHO command has two unrelated, independently functioning purposes:

- Command line echoing
- Message display

Command line echoing

When command line echoing is enabled, each command is displayed on stdout 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 command.

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.

Message display

If the ECHO command has a message (the whole command tail, 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, regardless of the applicable echoing state. Any display sent to stdout 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) or precede them with the escape character, or use the /X option of the SETDOS 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 **NUL** character cannot be included in *message.*
- If stdout is the console, after displaying *message* on the current line, the cursor will be moved to the beginning of the next line.
- If stdout is a file, the **CR LF** sequence will be appended to *message*.

    echo ``   (two consecutive back quotes), or
    echo.   (special syntax for compatibility with CMD).

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.

4.2.52 ECHOERR

Purpose: Display a message to the standard error device (stderr)

Format: ECHOERR *message*

> ***message***   Text to display.

See also: ECHO, ECHOS, ECHOSERR, ECHOX, and ECHOXERR.

Usage:

ECHOERR (like ECHO in message display mode) parses and expands ***message***, and displays it on stderr (usually the screen), instead of stdout. Even if stdout of a batch file is redirected or piped, ECHOERR will still display a screen message, unless stderr is redirected or piped (see Redirection). Any display sent to stderr 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) or precede them with the escape character, or use the /X option of the SETDOS 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 **NUL** character cannot be included in ***message***.
- If stderr is the console, after displaying ***message*** on the current line, the cursor will be moved to the beginning of the next line.
- If stderr is a file, the **CR LF** sequence will be appended to ***message***.

4.2.53 ECHOS

Purpose: Display a message to standard output (stdout) without a trailing carriage return / line feed

Format: ECHOS *message*

> ***message***   Text to display.

See also: ECHO, ECHOERR, ECHOSERR, ECHOX, ECHOXERR, SCREEN, SCRPUT, TEXT, and VSCRPUT.

Usage:

ECHOS, like ECHO in message display mode, parses, expands, and displays *message* on stdout. However, any display sent to stdout 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) or precede them with the escape character, or use the /X option of the SETDOS 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 **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.

4.2.54 ECHOSERR

Purpose: Display a message to the standard error device (stderr) without a trailing carriage return / line feed

Format: ECHOSERR *message*

              message Text to display.

See also: ECHO, ECHOS, and ECHOERR.

Usage:

ECHOSERR acts as a combination of ECHOS and ECHOERR. It parses and expands *message*, and displays it on stderr. However, any display sent to stderr 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) or precede them with the escape character, or use the /X option of the SETDOS 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 **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

4.2.55 ECHOX

Purpose: Display a message to standard output (stdout) without performing any variable expansion or redirection.

Format: ECHOX *message*

   message   Text to display.

See also: ECHO, ECHOERR, ECHOSERR, ECHOXERR, SCREEN, SCRPUT, TEXT, and VSCRPUT.

Usage:

ECHOX will echo the message text to STDOUT without doing any of the parser processing (variables, redirection, escaped characters, etc.).

Display rules

- The first space after the command name is ignored.
- Trailing spaces in *message* are ignored.
- The ASCII **NUL** character cannot be included in *message*.

ECHOX is useful for text output when you want to display some text that may have embedded special characters (like %, <. >, or |).

4.2.56 ECHOXERR

Purpose: Display a message to standard error (STDERR) without performing any variable expansion or redirection.

Format: ECHOXERR *message*

   message   Text to display.

See also: ECHO, ECHOERR, ECHOSERR, ECHOX, SCREEN, SCRPUT, TEXT, and VSCRPUT.

Usage:

ECHOXERR will echo the message text to STDERR without doing any of the parser processing (variables, redirection, escaped characters, etc.).

Display rules

- The first space after the command name is ignored.
- Trailing spaces in *message* are ignored.
- The ASCII **NUL** character cannot be included in *message*.

ECHOXERR is useful for text output when you want to display some text that may have embedded special characters (like %, <. >, or |).

4.2.57 EJECTMEDIA

Purpose: Eject removable media in the specified drive(s)

Format: EJECTMEDIA *drive* ...

Usage:

EJECTMEDIA will eject removable media, such as CD-ROMs, DVDs, removable USB drives, etc.

Example:

To eject the E: drive:

```
ejectmedia e:
```

See also LOADMEDIA.

4.2.58 ENDLOCAL

Purpose: Restore the saved disk drive, directory, environment, local alias and function lists, and special characters, and exports selected variables

Format: ENDLOCAL [/D] [*exportvar* ...]

              /D(ont restore)

See also: SETLOCAL.

Usage:

The SETLOCAL command saves the current disk drive, default directory, all environment variables, the alias and function lists, the directory stack (PUSHD), and the command separator, escape character, parameter character, decimal separator, and thousands separator. It does not save the user-defined function list or array variables. ENDLOCAL restores everything that was saved by the previous SETLOCAL command, except as described below.

If you have only global aliases and/or functions, SETLOCAL will copy them to a local list for the duration of the SETLOCAL. The matching ENDLOCAL will reset them to the global list. If you have both local and global aliases or functions, ENDLOCAL will only restore the local list (that was saved by SETLOCAL).

For example, this batch file fragment saves everything, removes all aliases so that user aliases will not affect batch file commands, changes the disk and directory, changes the command separator, runs a program, and then restores the original values:

```
setlocal
unalias *
cdd d:\test
setdos /c~
program ~ echo Done!
```

```
endlocal
```

SETLOCAL / ENDLOCAL may be nested within a single batch file up to 32 levels deep. You can also have multiple, separate SETLOCAL / ENDLOCAL pairs within a batch file, and nested batch files can each have their own SETLOCAL / ENDLOCAL. If you do not provide an ENDLOCAL in the batch file, TCC-RT will do it automatically when the batch file exits.

An ENDLOCAL is performed automatically at the end of a batch file, or when returning from a "GOSUB filename". If you invoke one batch file from another without using CALL, the first batch file is terminated, and an automatic ENDLOCAL is performed; the second batch file inherits the settings as they were prior to any SETLOCAL.

- **Exporting environment variables**

The environment variables whose names are specified in the ENDLOCAL command are exported. This means that their names and values from inside the SETLOCAL / ENDLOCAL will be placed into the restored environment, either adding variables, or possibly modifying them. In the example below, the variable TEST will have the value **abcd** after the ENDLOCAL is executed, regardless of what its value was, or even if it had not been previously defined:

```
setlocal
set test=abcd
endlocal test
```

The list of variables to export may contain wildcards. All variables matching the requested pattern will be exported.

- **Exporting current working directory**

See option /D below.

Options:

  /D   (Don't restore directory) Export the current directory: the original drive and directory saved by SETLOCAL will not be restored.

4.2.59 ENUMPROCESSES

  Purpose: Display the child processes for the specified process

  Format: ENUMPROCESSES *pid*

                pid   The parent process ID whose child processes you want to enumerate.

  Usage:

ENUMPROCESSES will show all of the active processes started by the specified process.

4.2.60 ENUMSERVERS

  Purpose: Enumerate the servers on the network

  Format: ENUMSERVERS [/Domain=*domain* /P[*n*] /Type=*xxx*] *server* ...

server   The machine name to match

/P(ause)
/Type=type

Usage:

The optional server name may contain wildcards, including regular expressions.

Option:

/Domain The NetBIOS name of the domain to enumerate. If you do not specify a domain, ENUMSERVERS will use the primary domain.

/P[*n*] Pause after each page. Your options at the prompt are explained in detail under Page and File Prompts. The /P option has an optional argument *n* that specifies the number of seconds to wait for a keystroke before continuing.

/Type   Return only servers of this type. If you do not specify /Type, TCC-RT will return all servers. Other 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

4.2.61 ENUMSHARES

Purpose: Enumerate the share names for the specified server

Format: ENUMSHARES [/A /D /F /I /P[*n*] /Q /R /U /V] *server**sharename*

server   The machine name
sharename   The sharename(s) to match. Sharenames may contain wildcards, including regular expressions.

/A(dmin)	/Q(ueues)
/D(isk)	/R(emarks)
/F (local path)	/U(ses)
/I(PC)	/V (devices)
/P(ause)	

Usage:

ENUMSHARES will show the share names of the specified type for a network server. ENUMSHARES will enumerate a single server; you cannot specify any wildcards in the server name.

Option:

/A	Display the admin shares (i.e., ADMIN$, C$, print$, etc.)
/D	Display the disk shares (default unless /F, /I, or /Q is set)
/F	The local path for the shared resource. For disks, this member is the path being shared. For print queues, this is the name of the print queue being shared.
/I	Display the shared IPC (interprocess communication).
/P[*n*]	Pause after each page. Your options at the prompt are explained in detail under Page and File Prompts. The /P option has an optional argument *n* that specifies the number of seconds to wait for a keystroke before continuing.
/Q	Display the shared print queues
/R	Display the optional comment about the sharename (in quotes)
/U	Display the current number of uses and the maximum allowed uses in the format "[n] [n]".
/V	Display the shared communication devices

4.2.62 ESET

Purpose: Edit an environment variable, alias or function definition

Format: ESET [/A /B /D /F /GL /LL /S /U /V] [/C *var1 var2*] *name*

name	The name of an environment variable, function or alias to edit.

/A(lias)	/K"::regex" (regex mask)
/B(atch variable)	/LL (local list)
/C(opy value)	/S(ystem variable)
/D(efault environment)	/U(ser variable)
/F(unction)	/V(olatile variable)
/GL (global list)	/W(indow)

See also: ALIAS, FUNCTION, SET, UNALIAS, UNFUNCTION, and UNSET.

Usage:

ESET allows you to edit environment variables, aliases or user-defined functions using line editing commands, or in a popup window.

Unless a specific data type is specified by one of the option switches **/A**, **/D**, **/F**, **/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.

If you are editing a typed environment variable (see SET /T), ESET will create a matching regular expression mask for the input.

Note: You cannot use ESET with GOSUB variables.

If you have enabled global aliases (see ALIAS), any changes made to an alias with ESET will immediately affect all other copies of TCC-RT which are using the same alias list. Similarly, if you have enabled global functions (see FUNCTION), any changes made to a function using ESET /F will immediately affect all other copies of TCC-RT which are using the same function list.

ESET will default to looking in the local alias or function list (if it exists). If the name isn't found, ESET will look in the global list (if it exists).

ESET supports filename completion and completion of internal variables and variable functions.

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.

Examples:

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
```

ESET allows you to edit environment variables, aliases or user-defined functions using line editing commands, or in a popup window.

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**, **/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.

If you are editing a typed environment variable (see SET /T), ESET will create a matching regular expression mask for the input.

Note: You cannot use ESET with GOSUB variables.

If you have enabled global aliases (see ALIAS), any changes made to an alias with ESET will immediately affect all other copies of TCC-RT which are using the same alias list. Similarly, if you have enabled global functions (see FUNCTION), any changes made to a function using ESET /F will immediately affect all other copies of TCC-RT which are using the same function list.

ESET will default to looking in the local alias or function list (if it exists). If the name isn't found, ESET will look in the global list (if it exists).

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.
/B	Edit a batch variable (%1 - %n). Only valid when TCC-RT is executing a batch file.
/C	Copy the value from an existing variable, alias, or function. The syntax is:

      ESET /c var1 var2

      where *var1* is the variable whose value you want to copy, and *var2* is the variable (new or existing) that you want to update.

/D	Edit a "default" variable in the registry (HKU\.DEFAULT\Environment).
/F	Edit a user-defined function.
/GL	Look for the alias or function in the global list
/K"::*regex*"	Regular expression mask for the input.
/LL	Look for the alias or function in the local list

/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	Open the alias list / environment / function list in a popup window and select the line to edit. You can search, edit, and delete entries in the window. If you include an argument after the /W option, the popup window will display only those entries that match the argument (including wildcards).
	ESET /W can be combined with a registry environment key (/S, /U, /D, /V) to edit the Windows registry environment values.

4.2.63 EVENTLOG

Purpose: Write a string to the Windows event log

Format: EVENTLOG [/S"*source*" /C*n* /E /I /W] *message*

message	The text to write.
source	The source for this message.

/C(ategory)   /S(ource)
/E(rror)   /W(arning)
/I(nformational)

See also: LOG.

Usage:

EVENTLOG posts messages to the Windows application event log. You cannot use the command separator character ([**&**]) or the redirection symbols (**| > <**) in an EVENTLOG message, unless you enclose the message in quotes or precede the special characters with the escape character.

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. If you are running Windows 7 or later, you will also need to be running an elevated session.

Options:

/C*n*   Set the event category. The value can be from 0-65535; 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 you use the /S option, it must be the first option on the EVENTLOG command line. 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.

4.2.64 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)   /S"source"
/D"description"   /T"type"

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).

If the last argument on the line is a single **(,** it is interpreted as the beginning of a command group. EVENTMONITOR will append the following lines (in a batch file) or prompt you for more input (at the command line) until it gets a closing **).**

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-RT thread is also performing I/O. You may need to use START or DETACH 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
_eventsource	The name of the source that wrote the event
_eventtype	The event type (see /T 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

4.2.65 EXCEPT

Purpose: Perform a command on all available files except those specified

Format: EXCEPT [/I"text" /N[em]] [(@*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 for details).
command	The command to execute, including all appropriate parameters and switches.

/I (match description)   /NM (no matches)
/Ne (no errors)

See also: ATTRIB and File Exclusion Ranges.

File Selection

Supports extended wildcards, ranges, multiple file names, and include lists. 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 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. Using internal commands with EXCEPT is not recommended. File exclusion ranges 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.

You may use wildcards to specify the files to exclude from the **command**. When you use EXCEPT on an LFN drive, you must quote any file names inside the parentheses which contain white space or special characters.

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 competed. 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 /Z, and the **/A:H** or **/H** (process hidden files) switches available in internal file processing 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 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 the last argument on the line is a single **(,** it is interpreted as the beginning of a command group. EXCEPT will append the following lines (in a batch file) or prompt you for more input (at the command line) until it gets a closing **)**.

Option:

/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 for details.

/Ne   Don't display an error message if EXCEPT can't find a matching file.

/NM   If EXCEPT cannot find any file matches for the arguments in the exception list, EXCEPT will not execute the command.

4.2.66 EXEC

Purpose: Replace the current TCC shell with another program

Format: EXEC *command [args ...]*

Usage:

EXEC will start the specified app in the TCC console, then exit TCC, leaving the app running.

4.2.67 EXIT

Purpose: Exit the current TCC-RT session

Format: EXIT [/B] [*value*]

         value   The numeric exit code to return.

         /B (exit from batch file)

Usage:

EXIT terminates the current copy of the command processor.

To close the session, or to return to the application that started the command processor, type:

```
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. The CANCEL and QUIT commands are generally more flexible for use in batch files.

4.2.68 EXPR

Purpose: Evaluate an expression and display the result on STDOUT

Format: EXPR *string : regex*   Returns position of regular expression *regex* in *string*
                     EXPR match *string regex*   Same as *string : regex*

EXPR substr *string pos len* Substring of *string*, *pos* starts at 1
EXPR index *string chars* Index in *string* (first character is 1) where anything in *chars* is found, or 0 if nothing matches
EXPR length *string* Length of *string*

arg1 [operator] arg2... This can be any arithmetic expression supported by @EVAL, or any conditional expression supported by IF /IFF.

See also: Conditional expressions, IFF, @IF.

Usage:

EXPR evaluates integer or string expressions, including pattern matching regular expressions.

If you have special characters (i.e., < > & |) on the line you must either enclose the entire expression in double quotes (EXPR will remove them before evaluating the expression) or escape them.

The regular expression match is always anchored (i.e., there is an implied leading ^). If the regular expression contains (**…**), and it matches at least part of *string*, **EXPR** returns that part of *string*; if there is no match, **EXPR** results in 0. If the regular expression doesn't contain (..), the result is the number of characters matched. MATCH performs the same operation as the colon operator.

Examples:

expr text : tex
3

expr text : (.*)
text

4.2.69 FALSE

Purpose: Returns a 0

Format: FALSE

Usage:

FALSE returns 0 and sets the ERRORLEVEL variable to 0.

4.2.70 FFIND

Purpose: Search for files by name or contents

Format: FFIND [/8 /A[[:][-]
rhsadecijopt /B /C /D[*list*] /E["*text*"] /F /G /H /I /I"text" /K /L /L*n* /M /N[dehjs] /O:[-]
acdeginorstuz /P /Q /R /S[[+]n] /[T|X]"xx" /U /V /Y /+n /-n] [@*file*] *file*...

list A list of disk drive letters (without colons).
file The file, directory, or list of files or directories to display.

/[+|-] skip matches /M (no footers)
/8 (UTF-8) /N(ot)
/A(ttribute select) /O(rder)

/B(are)	/P(ause)
/C(ase sensitive)	/Q(uiet)
/D(rive)	/R(everse search order)
/E (upper case)	/S(ubdirectories)
/E"xx" (regular expression)	/T"xx" (text search string)
/F (stop after match)	/TE"xx" (convert text search string to regular expression)
/G (goto directory)	/U (summary only)
/H (ignore binary files)	/V (verbose)
/I(gnore wildcards)	/X["xx"] (hex display / search string)
/I"text" (match description)	/Y (prompt to stop after match)
/K (no headers)	
/L(ine numbers or header/footer lines)	

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 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"Windows" ftp://ftp.microsoft.com/windows
```

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 and IFTP.

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.

FFIND sets three internal variables:

%_ffind_matches	The number of matches
%_ffind_files	The number of files found
%_ffind_errors	The number of errors

Options:

/8   The file is interpreted as UTF-8.

/A: Select only those files that have the specified attribute(s) set. See Attribute Switches for information on the attributes which can follow **/A:**.

You can specify **/A:=** to display a dialog to help you set individual attributes.

/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](). The regular expression must be contained in double quotes if it contains spaces, punctuation, or wildcard characters. See also **/T**. The maximum line length supported by /E is 16Mb.
/F	Stops the search after the first match.
/G	Change to the directory where the match was found (must be used with **/F**).
/H	Don't search for text in binary files. By default, this includes .exe, .sys, .dll, .zip, and .chm extensions. You can define your own list by setting the "BINARY_FILES" environment variable. For example, to ignore .exe, .sys, and .dll files:

```
BINARY_FILES=.exe;.sys;.dll
```

/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]() 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 line for text searches.
/L	Include the line number for each text line displayed for text searches. 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, whichever comes first.

/Ln	The number of leading and trailing lines to display on a text search match. Each successive group of lines in a file will be separated by a "----" header.
/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.

A **/N** with one or more of the following arguments has an alternate meaning:

d	Skip hidden directories
e	Don't display errors.
h	No headers
j	Skip junctions
s	Don't display the summary.

/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.
/Q	Don't display any output, but set the FFIND internal variables (%_ffind_matches, %_ffind_files, and %_ffind_errors).
/R	Only meaningful when used in text searches 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 source directory and all of its subdirectories. If you don't specify a path with the *file* to search, FFIND will default to starting in the current directory.

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.

If you specify a **+** followed by a number after the /S, FFIND will not search for files until it gets to that depth in the subdirectory tree. For example, if you have a directory tree \a\b\c\d\e, /S+2 will not find anything in \a or \a\b.

/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.

/TE*"text"*   Converts a text string (see /T above) to a regular expression and then does a regex search. You don't need to learn regular expressions, and /TE will run 10x faster than /T. The only limitation is the maximum line length in the file must be < 16Mb.

/U   Display only the summary.

/V   Show every matching line on a text search. 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 **/E** or **/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.

4.2.71 FILELOCK

Purpose: Show processes locking a file

Format: FILELOCK]/C /F] *filename*

      /C(lose)
      /F(orce close)

Usage:

FILELOCK returns a list of the processes with a lock on the specified file, and optionally closes them to free the file.

Options:

  /C   Requests the process(es) close.

  /F   Like TASKEND /F, forces the process(es) to close.

4.2.72 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)

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 the last argument on the line is a single **(,** it is interpreted as the beginning of a command group. FIREWIREMONITOR will append the following lines (in a batch file) or prompt you for more input (at the command line) until it gets a closing **).**

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-RT thread is also performing I/O. You may need to use START or DETACH 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 (this may have special characters like **&** in the name, so you may need to use double quotes around the variable name to prevent TCC-RT from parsing the special characters)
_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.

4.2.73 FOLDERMONITOR

Purpose: Monitor folder and/or file creation, modification, and deletion

Format: FOLDERMONITOR [/C [*folder*]]
FOLDERMONITOR /Wn /S *folder* /I"*file*" /E"*file*" /U CREATED DELETED MODIFIED RENAMED *n command*

folder	Folder (directory) 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(lear)	/S(ubdirectories)
/E(xclude)	/U(nlocked file)
/I(include)	/W(ait)

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.

If the last argument on the line is a single **(,** it is interpreted as the beginning of a command group. FOLDERMONITOR will append the following lines (in a batch file) or prompt you for more input (at the command line) until it gets a closing **).**

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-RT thread is also performing I/O. You may need to use START or DETACH 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
_foldertime	System time when the change occurred.

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.
/U	Don't set the trigger until the file is unlocked. Not compatible with CREATED, because FOLDERMONITOR will always get the notification before the file is accessible.
/Wn	Wait for *n* milliseconds before processing the file / directory change. This is useful if you have a lot of actions occurring in a short period and you only care about the last one. If you specify /Wn, it must be the first argument in the FOLDERMONITOR command.

4.2.74 FONT

Purpose:	Change the console font
Format:	FONT [/F*family* /N*name* /W*n* /X*n* /Y*n*]

/F(ont family)	/X (width)
/N (face name)	/Y (height)
/W(eight)	

Usage:

This command will only affect stand-alone TCC-RT console windows. (You can change the font in Take Command tab windows using Configure Take Command / Tabs.)

Options:

/F	The font family:
	decorative
	dontcare
	modern
	roman
	script
	swiss
/N	The font face name.
/W	The font weight (100 - 1000, on multiples of 100). The normal weight is 400; bold is 700.
/X	The maximum width of a character, in logical units.
/Y	The maximum height of a character, in logical units.

4.2.75 FOR

Purpose:	Repeat a command for several values of a variable
Format:	File and string mode

FOR [*range*...] [/l"text"] [/A:[[-|+]rhsadecijopt /D /F ["*options*"] /H /Nj /O:[-]adegnrstu /R [*path*] [/T"*delimiters*"] /W] %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 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)	/N (defaults)
/D(irectories only)	/O:... (Order)
/F(ile parsing)	/R(ecursive)
/H(ide dots)	/T (delimiter list)
/I description range	/W(ildcards)
/L (counted loop)	

File Selection

Supports attribute switches, extended wildcards, ranges, multiple file names, and include lists.

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 for details.

Usage:

FOR begins by creating a *set*. It then 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.

The first three sections below (Working with Files, Working with Text, and Retrieving Text from Files) describe the FOR command and the enhancements to it which are included in TCC-RT. The sections on Parsing Text from Files and Counted FOR Loops describe features added for compatibility with CMD. The sections Directory Recursion and Output Redirection warn of special considerations. The section entitled Other Notes contains information you may need if you use any aspect of the FOR command extensively.

FOR sets two internal variables:

%_for_files	The number of files processed
%_for_errors	The number of errors

If the Duplicate CMD Bugs configuration option is set, TCC-RT will emulate undocumented CMD behavior when FOR *set* arguments are split across multiple lines. For example:

```
for %a in (
one
two
three
) do (
echo %a
)
```

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 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, as well as extended parent directory names, *e.g.*, . . . *.*txt* to process all of the *.TXT* files that are contained in the directory 2 levels above the current directory.

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-RT 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. 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, 2018, and of the file ABC.LST regardless of its timestamp:

```
for /[d12-4-2018,+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: 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-RT 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 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. 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-RT, for example, one of the characters <"[]|>, it may have undesirable effects. You may use the /X option of SETDOS 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:** (or @CLIP0: - @CLIP9:) as the filename, FOR will read any text available from the Windows clipboard. See Redirection and Piping for more information on these features.

See @file list 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 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, can be cumbersome and inflexible. For a more powerful method, use FOR with @filename as the *set* to retrieve each line from the file, as described in the previous section, and use variable functions like @FIELD, @INSTR, @LEFT, @RIGHT, and @WORD to parse the line (see Variable Functions 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 TEST.TXT in **%d**, the second in **%e**, and the third in **%f**:

```
for /f "tokens=1,2,3" %d in (test.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, tabs and commas 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 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-RT, as TCC-RT 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. In most cases, you will find the DO command more useful for performing counted loops.

In a counted FOR command, the *set* is made up of numeric values instead of text or file names. To begin a counted FOR command, you must use the /L option and then include three values, separated by commas, in *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 (a sequence of 0-9 digits) 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] specifies that the search should recursively process subdirectories. If you specify a directory name immediately after [/R], 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 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 specifications will be ignored.
- The word DO is unnecessary but accepted. Do not confuse it with the completely unrelated DO command.
- If the name of the FOR variable **var** is a single character, for compatibility with CMD, 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, 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, ESET, or UNSET 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 setting, then erase the PATH variable completely when FOR is done.

- **Command** may also use the FOR variable with the special syntax of CMD described in Special syntax for CMD compatibility.

- The following example uses FOR with variable functions to delete the *.BAK* files for which a corresponding *.TXT* file exists in the current directory (this should be entered on one line):

```
for %file in (*.txt) del %@name[%file].bak
```

The above command may not work properly on an LFN drive, because the returned *FILE* variable might contain white space. To correct this problem, you need two sets of quotes, one for DEL and one for %@NAME:

```
for %file in (*.txt) del "%@name["%file"].bak"
```

- You can use command grouping 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 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 CALLing another batch file.

- One unusual use of FOR is to execute a collection of batch files or other commands with the same parameter. For example, you might want to have three batch files all operate on the same data file. The FOR command could look like this:

```
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 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 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.

You can specify **/A:=** to display a dialog to help you set individual 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.

/H   Suppresses the assignment of the "." and ".." directories to the FOR variable when directories are explicitly included using the /A: option.

/I"*text***"**   Select filenames by matching text in their descriptions. See Description Ranges.

/L   Interpret the three values in **set** as the **start**, **step**, and **end** values of a counted loop. See Counted FOR Loops.

/Nj   Don't recurse into symlinks or junctions (see /R).

/O:...   Sort the files before processing.

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, instead of numerically when numeric substrings are included in the name or extension.
  d   Sort by date and time (oldest first); also see **/T:acw**
  e   Sort by extension
  g   Group subdirectories first, then files
  r   Reverse the sort order for all options
  s   Sort by size
  t   Same as **d**
  u   Unsorted

The /O:... option saves all of the matching filenames and then performs the requested operation. This avoids the potential problem of processing files more than once.

/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 ("). /R supports Windows shell folder names; see CDD for details.

/T"*text***"** Specify the delimiters to be used when parsing a string set.

/W   The FOR set is to be processed as filenames, even if no wildcards are detected. (This is useful if you want to use regular expressions with FOR.)

4.2.76 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.

Usage:

A colon [**:**] is required after each drive letter.

If the volume serial number is available, it will appear after the drive label or name.

FREE supports OpenAFS names.

Example:

Display the drive status of drives C and D :

    `free c: d:` names.

4.2.77 FSEARCH

Purpose: Search files for the specified text

Format: FSEARCH [/= /+*n* /-*n* /8 /B /C /E"*regex*" /F /G /H /I /L /N[dehjs] /Q /S[[+] *n*] /T"*text*" /U /V /Y /Z] [*path*] *filename*

Usage:

FSEARCH is a modern replacement for the aged FFIND. New features will only be implemented in FSEARCH, not FFIND.

FSEARCH will automatically determine the file type (ASCII, UTF8, or UTF16). You can also tell FSEARCH to assume UTF8 files with the /8 option. (This is slightly faster because FSEARCH doesn't have to pre-scan the files trying to determine the encoding.)

You can use **TCC** extended wildcards in the search string. 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:

    `fsearch /c /t"Ju[nl][ey]" *.txt`

You can also search using regular expressions using the **/E** option. See Regular Expression Syntax for supported expressions.

When you use FSEARCH on an LFN drive, you must quote any file names which contain white space or special characters.

FSEARCH sets three internal variables:

_fsearch_errors - Errors when running FSEARCH (i.e., file/path not found, file locked, access denied, etc.)

_fsearch_files - The number of files containing one or more matches

_fsearch_matches - The total number of matches

Options:

/=   Display the FSEARCH dialog

/+*n*   Skip the first *n* matches

/-*n*   Stop after *n* matches

/8   Instead of scanning the files for their type, they are assumed to be UTF8 (this is a little faster).

/B   Only display filenames (no header or footer or summary or matching lines)

/C   Match case

/E"..." Regular expression search

/F   Stop after first match (overrides /V)

/G   Change to the directory containing the first matching file (also sets /F and overrides /V)

/H   Don't search for text in binary files. By default, this includes .exe, .dll, .sys, .chm, .zip, .pdb, .pch, .obj, .tar, .com, and .ewriter. You can define your own list by setting the "BINARY_FILES" environment variable.

For example, to ignore .exe, .sys, and .dll files:

```
BINARY_FILES=.exe;.sys;.dll
```

/I   Used with /T to tell FSEARCH to ignore wildcard characters (*, ?, and [...]).

/L   Display line numbers for matching text

/N... Disable options:

D   Don't scan hidden subdirectories
E   Don't display errors
H   No header
J   Skip junctions
S   No footer (summary)

/Q   Don't display any output. The internal variables (see below - _fsearch_errors, _fsearch_files, and _fsearch_matches **are** set).

/S   Search subdirectories of the specified (or default) path.

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.

If you specify a + followed by a number after the /S, FFIND will not search for files until it gets to that depth in the subdirectory tree. For example, if you have a directory tree \a\b\c\d\e, /S+2 will not find anything in \a or \a\b.

/T"..."   Search for the matching text. Supports TCC wildcards (?, *, and [...]).

/U   Only display summary line (no filenames or matching lines; overrides /V)

/V   Display all matching text (FFIND defaults to only displaying the first match in each file)

/Y   Display a "Continue Y/N" prompt after displaying each match

/Z   Highlight the matched text

4.2.78 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)   /U(ser)
/R(ead from file)

See also: ASSOC, ASSOCIATE, and Executable Extensions.

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. 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.

/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. 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 (**:**).

/U Display or set the association in HKCU\Software\Classes.

4.2.79 FUNCTION

Purpose: Create, modify or display user-defined variable functions

Format: Display mode:
FUNCTION [/GL /LL /P] [*wildname*]

Direct definition mode:
FUNCTION [/GL /LL] *name*[=]*definition*

Definition file mode:
FUNCTION [/G /GL /L /LL /O /Z] /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)   /P(ause)
/GL (global list)   /R(ead file)
/L(ocal)   /Z (overwrite list)
/LL (local list)

See also: <u>UNFUNCTION</u> and <u>ESET</u>.

Usage:

- ▸ <u>Overview</u>
- ▸ <u>Displaying Functions</u>
- ▸ <u>Defining Functions</u>
- ▸ <u>Deleting Functions</u>
- ▸ <u>Local and Global Functions</u>
- ▸ <u>Saving and Reloading Your Functions</u>
- ▸ <u>Warnings</u>

Overview

FUNCTION allows you to create or display user-defined variable functions that can be used anywhere <u>Variable Functions</u> can be used. User-defined functions are powerful alternatives to <u>subroutines</u>.

Displaying Functions

If you invoke the FUNCTION command with no parameters, it will display the current function list (the local function list if you have set local functions in *TCMD.INI* or the TCC-RT startup command line; otherwise the global 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 <u>/P</u> 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 <u>ESET</u> /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-RT interprets it to mean "all arguments, from parameter ***n*** to the end." If ***n*** is not specified, it has a default value of **1**, so **%\$** means "all arguments passed to the function."

The parameter **%-n$** means "the arguments 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 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 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 local and/or global lists.

With a local function list, any changes made to the functions will only affect the current copy of TCC-RT. They will not be visible in other shells or other sessions.

With a global function list, all copies of TCC-RT 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-RT.

You can control the type of function list to use with the /GL and /LL options in FUNCTION, Local Functions and Global Functions configuration options, with the **/L** and **/LF** options of the START command, and with the **/L** and **/LF** startup options.

If you don't specify /GL or /LL, TCC-RT will first look for functions in the local list. If there is no local list or the function is not found, TCC-RT will search the global list (if it exists).

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-RT 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 second shell will affect only that shell. If you want changes made in the second 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-RT 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-RT.

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 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. If you already have a global function list (for example, in another TCC-RT instance or in SHRALIAS), FUNCTION will not do the conversion.
/GL	Read from and write to the global function list. If you have both local and global function lists defined and do not specify /GL, FUNCTION will default to using the local list.
/L	Switch from a global to a local function list.
/LL	Read from and write to the local function list.
/O	Don't overwrite existing values (only valid in combination with /R).
/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. 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
```

FUNCTION /R definitions can span multiple lines in the file if each line, except the last, is terminated with an [Escape Character](#).

If there is no filename parameter and input is redirected, FUNCTION /R will read from stdin.

/Z   Overwrite the existing function list with the contents of the specified file (must be used with /R). FUNCTION /R /Z is 20x faster than FUNCTION /R, because it doesn't have to check for existing functions and append new functions to the end of the list.

4.2.80   GLOBAL

Purpose: Execute a command in the current directory and its subdirectories

Format: GLOBAL [/H /I /J /N /P /Q /S[+]*n*] *command*

command   The command to execute, including parameters and switches.

[/H(idden directories)](#)   [/P(rompt)](#)
[/I(gnore exit codes)](#)   [/Q(uiet)](#)
[/J (only junctions)](#)   [/S(ubdirectory depth)](#)
[/N(o junctions)](#)

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[]](#), [@full[]](#), 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](#) 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](#) 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](#) 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-RT 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.
/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 "\a\b\c\d\e", /S2 will only go to the "a", "b", and "c" directories.
	If you specify a **+** followed by a number after the /S, GLOBAL will not execute *command* until it gets to that depth in the subdirectory tree. For example, if you have a directory tree \a\b\c\d\e, /S+2 will not execute *command* in \a or \a\b.

4.2.81 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, GOTO, and RETURN.

Usage:

GOSUB can only be used in batch files.

TCC-RT 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, and cannot be included a command group. Case differences are ignored when matching labels. The subroutine must end with a RETURN statement.

The subroutine is invoked with a GOSUB command from another part of the batch file. After the RETURN, processing will continue with the command following the GOSUB command. For example, the following batch file fragment calls a subroutine which displays the directory and returns:

```
echo Calling a subroutine
gosub subr1
echo Returned from the subroutine
quit
:subr1
dir /a/w
return
```

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 call.

If you append a * to the last variable name in the parameter list on the label line, it will be "greedy", and all remaining variables will be assigned to it. For example:

```
gosub sub1 one two three four five
...
:sub1 [arg1 arg2 arg3*]
```

arg3 will be assigned "three four five".

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, ESET, or UNSET commands, and cannot be used with the DEFINED test of IF, IFF, or @IF.

You cannot use SET within a subroutine to change the value of a GOSUB variable. If you attempt to do so, the SET command will set the standard environment variable of the same name, not the GOSUB variable, but this value will be "masked" by the GOSUB variable and will remain inaccessible until the subroutine ends.

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 from a GOSUB while inside a DO loop.

If TCC-RT reaches the end of the batch file while inside a subroutine, it will automatically return to the command after the GOSUB, just as if an explicit RETURN command had been included as the last line of the file.

Subroutines can be nested.

See also: user-defined functions.

4.2.82 GOTO

Purpose: Branch to a specified line inside the current batch file

Format: GOTO [/l] *label*

         label   The batch file label to branch to.

         /l(FF and DO continue)

See also: GOSUB, CALL.

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, and cannot be included in a command group. 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 statements and DO / ENDDO loops unless you use **/I**. This means that a normal GOTO (without **/I**) may not branch to any label that is between an IFF and the corresponding ENDIFF or between a DO and the corresponding ENDDO.

For compatibility with CMD, the command

```
GOTO :EOF
```

will end processing of the current batch file if the label :EOF does not exist. However, this is less efficient than using the QUIT or CANCEL command to end a batch file.

Option:

/I   Prevents GOTO from canceling IFF statements and DO loops. Use this option only if you are absolutely certain that your GOTO command is branching entirely within any current IFF statement **and** any active DO / ENDDO block. Using **/I** under any other conditions will cause an error later in your batch file.

You cannot branch into another IFF statement, another DO loop, or a different IFF or DO nesting level, whether you use the **/I** option or not. If you do, you will eventually receive an "unknown command" error (or execution of the UNKNOWN_CMD alias or plugin) on a subsequent ENDDO, ELSE, ELSEIFF, or ENDIFF statement.

4.2.83   GZIP

Purpose: Create or update .gz (GZIP) archives

Format: GZIP [/A:[[-][+]rhsdaecjot] /A /En /Ln /M /O:[-]adegnrstu /Q /V] *gziparchive* [*@file*] *file*

gziparchive	The gzip file to work with
file	The files to be added to the gzip file

/A:... (attribute switch)	M(ove)
/A(dd)	/O:... (sort order)
/E (method)	/Q(uiet)
/Ln (compression level)	/V(iew)

See also UNGZIP.

File Selection

Supports attribute switches, extended wildcards, ranges, multiple file names, and include lists.

Usage:

GZIP is compatible with the archives created by the Linux / UNIX gzip utility, and supports RFC 1952. GZIP is normally used for compressing a single file. If you need to compress multiple files, you should use the ZIP or TAR commands.

You can specify a pathname for g*ziparchive*. If you don't provide an extension, and the filename as entered doesn't exist, GZIP adds ".gz". If you don't specify an operation, GZIP will default to Add.

Option:

/A:...   Select only those files that have the specified attribute(s) set. See Attribute Switches for information on the attributes which can follow /A:. Do not use /A: with ***@file*** lists. See @file lists for details.

You can specify **/A:=** to display a dialog to help you set individual attributes.

/A   Add the specified file to the archive. (This is the default.)

/En   Set the compression method (0=deflate, 1=lzw). The default is 0.

/Ln   Set the compression level (1 - 6, where 6=maximum compression). The default is 4.

/M   Delete the file from the disk after adding them to the gzip file.

/O:...   Sort the files before processing.

You may use any combination of the sorting options below. If multiple options are used, the files 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, instead of numerically when numeric substrings are included in the name or extension.
d	Sort by date and time (oldest first); also see **/T:acw**
e	Sort by extension
g	Group subdirectories first, then files
r	Reverse the sort order for all options
s	Sort by size
t	Same as **d**
u	Unsorted

/Q   Don't display the file being compressed.

/V   View the contents of the .gz file (date, time, and filename). If the file was compressed with lzw, it will not have a header, so it cannot be viewed.

4.2.84 HASH

Purpose: Display a hash value for the specified file(s)

Format: HASH [/
[*range*] /A:xxx /E /L /S /CKSUM /CRC32 /MD5 /SHA1 /SHA256 /SHA384 /SHA512]
filename ...

File Selection

Supports extended wildcards, ranges, multiple file names, and include lists.

Usage:

The HASH command generates a file's hash value using the specified algorithm. The hash value is a unique value corresponding to the content of a file. If two files have the same hash value and they're using SHA256, SHA384, or SHA512, then they have the same content (regardless of their file names or locations).

HASH will default to SHA256.

Example:

Return a SHA256 hash (in lower case) for the file TCMD.EXE:

```
hash /sha256 /L tcmd.exe
```

Option:

/A:	Select only those files that have the specified attribute(s) set. See Attribute Switches for information on the attributes which can follow **/A:**. Do not use /A: with @file lists. See @file lists for details. You can specify **/A:=** to display a dialog to help you set individual attributes.
/E	Ignore errors (i.e., file not found)
/L	Display hash in lower case
/S	Hash matching files in the current directory and subdirectories
/CKSUM	A Linux cksum-compatible CRC32 10-digit decimal number. CKSUM is not considered secure or unique.
/CRC32	A CRC-32 hash value is an 8-digit hexadecimal number. CRC-32 is not considered secure or unique.
/MD5	An MD-5 hash value is a 32-digit hexadecimal number. MD-5 is not considered secure.
/SHA1	A SHA-1 hash value is a 40-digit hexadecimal number. SHA-1 is no longer considered secure.
/SHA256	A SHA-256 hash value is a 64-digit hexadecimal number
/SHA384	A SHA-384 hash value is a 96-digit hexadecimal number

/SHA512   A SHA-512 hash value is a 128-digit hexadecimal number

4.2.85 HEAD

Purpose: Display the beginning of the specified file(s)

Format: HEAD [/A:[[-][+]rhsadecijopt] /B /C*n* /I"text" /N[+]*n* /O:[-]acdeginorstuz /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 for details).

             /A: (Attribute select)   /O:... (Order)
             /B(ell)   /P(ause)
             /C (number of bytes)   /Q(uiet)
             /I"text" (match description)   /V(erbose)
             /N(umber of lines)

See also: LIST, TAIL, and TYPE.

File Selection

Supports attribute switches, extended wildcards, ranges, multiple file names, and include lists.

Internet: Can be used with FTP/HTTP Servers, e.g.

```
head "https://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 (.*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.

HEAD sets two internal variables:

    %_head_files   The number of files displayed
    %_head_errors   The number of errors

HEAD will recognize Unicode (UTF-16) files based on either a BOM or specific UTF-16 sequences at
the beginning of the file. HEAD will recognize UTF-8 files based on either a BOM or UTF-8 extended
characters within the first 2K of the file.

FTP Usage

HEAD can also display files on <u>FTP/HTTP Servers</u>. For example:

```
head ftp://ftp.microsoft.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 <u>Attribute Switches</u> for information on the attributes which can follow **/A:**. Do not use /A: with @file lists. See <u>@file lists</u> for details.

You can specify **/A:=** to display a dialog to help you set individual attributes.

/B Ignore bell (ASCII 7) characters.

/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 <u>@file lists</u> for details.

/N+n Skip the first **n** lines.

/N n Display **n** lines. The default is 10.

/O:... Sort the files before processing.

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, instead of numerically when numeric substrings are included in the name or extension.
    c   Sort by compression ratio
    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 description
    o   Sort by owner

	r	Reverse the sort order for all options
	s	Sort by size
	t	Same as **d**
	u	Unsorted
	z	Same as **s**

/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.

4.2.86 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
	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)

See also: Conditional expressions, IFF, @IF.

Usage:

IF is usually used only in aliases and batch files. It is always followed by a ***condition*** (see Conditional expressions), and then a ***command***. First *condition* 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 command operators (**||** or **&&**).

Do not use IF with multi-line TCC-RT commands like DO (unless you use the single-line variant of DO).

The **IF ... ELSE ...** syntax of CMD is also supported:

```
IF [/I] condition (command1) ELSE (command2)
```

The commands to be executed must be enclosed in parentheses (as in a command group). If ***condition*** is TRUE, ***command1*** is executed, if FALSE, ***command2*** is executed. **Note:** this syntax is much less powerful than the IFF command, which is recommended.

If the last argument on the line is a single **(,** it is interpreted as the beginning of a command group. IF will append the following lines (in a batch file) or prompt you for more input (at the command line) until it gets a closing **).**

When an IF test fails, the remainder of the command is discarded. Whether TCC-RT 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 Bugs** configuration option. CMD 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. The IF behavior is different when **DuplicateBugs** is **YES** in a command group in a batch file. If there are multiple command lines in the command group, a failed IF will only ignore the remainder of the commands on that line. The commands on the subsequent lines in the command group will still be executed.

For example, if Duplicate CMD Bugs 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 Bugs 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. It has no effect in TCC-RT, since all string comparisons are case-insensitive unless you specify a case-sensitive test (EQC).

4.2.87 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	Conditional expressions
commandset1	One or more commands to execute if ***condition1*** is TRUE
commandset2	One or more commands to execute if ***condition1*** is FALSE, but ***condition2*** is TRUE.
commandset3	One or more commands to execute if both ***condition1*** and ***condition2*** are FALSE.

See also: IF and @IF.

Usage:

IFF is similar to IF, but it can perform one ***commandset*** when a conditional expression 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 expressions, and execute the ***commandset*** associated with the first TRUE conditional expression, 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 :

- 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, 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.

If the last argument on the line is a single **(,** it is interpreted as the beginning of a command group. IFF will append the following lines (in a batch file) or prompt you for more input (at the command line) until it gets a closing **).**

Notes

Be sure to read the cautionary notes about GOTO and IFF under the GOTO command before using a GOTO inside an IFF statement.

If you pipe data to an IFF, the data will be passed to the command(s) following the IFF, not to IFF itself.

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`
```

4.2.88 IFTP

Purpose: Open or close an FTP / FTPS / SFTP session

Format: IFTP [/= /S
command /C /EP /IPv6 /K="*key*" /N /P*n* /PR="n" /Q /R /V /V=*hostname*] /Z[*n*]] ["ftp://
[*user*[:*password*]@]*server*[/*path*][:*port*]"]

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.
port	Port number.

/C(lose)	/Q(uiet)
/EP (extended passive)	/R(econnect)
/IPv6	/S(end)
/K (SSH public key)	/T(cp keep-alive)
/N(o paths)	/V(erbose)
/P(assive)	/V=hostname
/PR="n" (port range)	/Zn (zlib)

Usage:

Most file processing commands and functions in TCC-RT 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 repeatedly starts an FTP session, performs an individual operation, and closes the FTP session, until the command or function is finished.

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@ftp.myserver.com/dir1

For an FTPS connection, use something like:

    iftp ftps://user:pwd@ftp.myserver.com/dir1

This command tells IFTP to open an FTP/FTPS session with the server **myserver.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. Double quotes are required if there are spaces or special characters in the filename. If you specify a password of *****, you will be prompted to enter the password (which will 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](#) 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-RT 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 **microsoft.com**:

```
dir "ftp://ftp.microsoft.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-RT 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, _iftps, and _isftp variables.

IFTP and the other FTP features of TCC-RT 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-RT expects, commands may not work as you intend. We urge you to test each server you use with nondestructive commands like DIR before you try to copy or delete files, create or remove directories, etc.

IFTP will try to preserve timestamps when transferring files. The MDTM command is used when downloading, and the MFTM command is used when uploading. If the FTP server does not support these commands, the current date/time will be used for the timestamp.

Before you can use IFTP, you must establish the necessary connection to the Internet.

See FTP Servers for additional information on formatting and usage of FTP and FTPS references.

Options:

/C   Use this switch, with no URL, to close an IFTP session (see the example above).

/EP   Use Extended Passive mode. (Works with FTP and FTPS, but not SFTP.)

/IPv6[=*0*|*1*|*2*]   By default, IFTP expects an IPv4 address for the local and remote host, and will create an IPv4 socket. The /IPv6 option tells IFTP to use IPv6 instead. (Works with FTP, FTPS, and SFTP connections.) When set to 0, IFTP will use IPv4 exclusively. When set to 1, IFTP will use IPv6 exclusively. To instruct IFTP to prefer IPv6 addresses, but use IPv4 if IPv6 is not supported on the system, this setting should be set to 2. If you don't specify /IPV6, IFTP will set this value to 0. If you specify /IPV6 with no explicit value, IFTP will set the value to 1.

/K="..."   The CA signed client public key used when authenticating (SSH only). When authenticating via public key authentication this setting may be set to the CA signed client's public key. This is useful when the server has been configured to trust client keys signed by a particular CA. For example:

```
/K="SignedSSHCert=ssh-rsa-cert-v01@openssh.com
AAAAB3NzaC1yc2EAAAADAQABAAAB..."
```

The algorithm such as ssh-rsa-cert-v01@openssh.com in the above string is used as part of the authentication process. To use a different algorithm simply change this value. For instance all of the following are acceptable with the same signed public key:

- ssh-rsa-cert-v01@openssh.com AAAAB3NzaC1yc2EAAAADAQABAAAB...
- rsa-sha2-256-cert-v01@openssh.com AAAAB3NzaC1yc2EAAAADAQABAAAB...
- rsa-sha2-512-cert-v01@openssh.com AAAAB3NzaC1yc2EAAAADAQABAAAB...

/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.

/PR="n" When using active mode, IFTP uses any available port to listen to incoming connections from the server. You can override this behavior by setting /PR (PortRange) to a value containing the range of ports the class will be listening to. The range is provided as *start-end*, for instance: "1024-" stands for anything higher than 1024, "1024-2048" stands for ports between 1024 and 2048 inclusive, "4000-4010, 50000-50010" stands for ports between 4000 and 4010 or between 50000 and 50010. (Works with FTP and FTPS, but not SFTP.)

/Q   Turn off the display of the conversation with the FTP server.

/R   Automatically reconnect if the FTP server times out.

/S   Allows you to send commands directly to an FTP server. The connection must have already been opened by a previous IFTP command.

/T=*n*   If this is set, the socket's keep-alive option is enabled, and TCP keep-alive packets will be sent periodically to maintain the connection. *n* is the inactivity time in seconds before a TCP keep-alive packet is sent.

/V   Display the dialog with the FTP server while opening the connection. This can be useful for debugging connection problems.

/V=*hostname*   Sends the HOST command to the server. The HOST command allows FTP processes to specify which virtual host to connect to for a server-FTP process that is handling requests for multiple virtual hosts on a single IP address. When this option is set, the HOST command is sent to the server prior to authenticating.

/Zn   Use Zlib compression. You can optionally set the compression level (0-9; the default is 7). Zlib compression must be enabled on the server, and will only work with FTP and FTPS connections (not SFTP).

4.2.89 INKEY

Purpose: Get a single keystroke from the user and store it in an environment or array variable

Format: INKEY [/= /C /D /K"*keys*" /P /M /T /W*n* /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	Clear buffer	/P	Password
/D	Digits only	/T	Countdown timer
/K	valid keystrokes	/W	Wait
/M	Mouse buttons	/X	no carriage return

See also: INPUT.

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 or array variable. It is normally used in batch files and aliases to get a menu choice or other single-key input. Along with the INPUT command, INKEY allows great flexibility in reading input from within a batch file or alias.

If *prompt* is included in an INKEY command, it is displayed while INKEY waits for input.

The following command 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. You can supply a list of valid keystrokes with the /K option.

Numeric input may be entered in either decimal format (a sequence of 0-9 digits) 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 for scan codes.

When the /M option enables recognition of mouse buttons, (and /W is not specified), the variable is set to a single character with one of the codes below:

button	code
left	240
middle	498
right	497

You can get the screen position of the last mouse click with the _xmouse and _ymouse internal variables.

To test for a non-printing value returned by INKEY use the @ASCII function to get the numeric value of the key, or convert the expected value of the code to a code using @CHAR. For example, to test for **Esc**, which has an ASCII value of 27 or a left mouse button:

```
inkey Enter a key:  %%key
if "%@ascii[%key]" == "27" echo Esc pressed
if %key EQ %@char[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 command.

If you don't enter any arguments, INKEY will display its command dialog.

INKEY works within the command line window. If you prefer to use a dialog for user input, see the MSGBOX and QUERYBOX commands.

Options:

/=	Display the INKEY command dialog to help you set the command line options. The /= option can be anywhere on the line; additional options will set the appropriate fields in the command dialog.
/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. You can use the /C option by itself if you want to clear the keyboard buffer without setting a variable.
/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 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-RT will echo the keystroke if possible, beep, move the cursor back one character, and wait for another keystroke.

/M	Accept mouse button clicks. This is enabled only if Windows' Quick Edit is disabled (alt-space -> Properties -> Options).
/P	Prevents INKEY from echoing the character.
/T	Display a countdown timer (/W*n* is also required).
/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.

4.2.90 INPUT

Purpose: Get a string from the keyboard and save it in an environment or array variable

Format: INPUT [/= /C /D /E["*default*"] /K"*keys*" /L*max*[:*min*] /N /P /T /W*n* /X] [*prompt*] *%%*
varname

        prompt   Optional text that is displayed as a prompt.
        varname   The variable that will hold the user's input.

/C(lear buffer)   /N(o colors)
/D(igits only)   /P(assword)
/E(dit)   /T (countdown timer)
/K(eys)   /W(ait)
/L(ength)   /X (no carriage return)

See also: SET, INKEY, KEYSTACK, MSGBOX, and QUERYBOX.

Usage:

INPUT optionally displays a prompt, then waits for your entry and stores it in an environment or array variable. INPUT is normally used in batch files and aliases to get multi-character input (for single keystroke input, see INKEY).

INPUT works within the command line window. If you prefer to us a dialog for user input, see the MSGBOX and QUERYBOX commands.

If ***prompt*** text is included in an INPUT command, it is displayed while INPUT waits for input. Standard command line editing keys may be used to edit the input string as it is entered. If you use the **/P** password option, INPUT will echo asterisks instead of the keys you type.

INPUT returns when you press carriage return. 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.

INPUT supports regular expressions for the mask (/K"xxx"). You must prefix the regular expression with **::** - for example:

```
    input /k"::^[0-9]$" Enter a number: %%number
```

Numeric input may be entered in either decimal format (a sequence of 0-9 digits) 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 command.

You can pipe 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-RT used to start the pipe.

If you don't enter any arguments, INPUT will display its command dialog.

Options:

/=	Display the INPUT command dialog to help you set the command line options. The /= option can be anywhere on the line; additional options will set the appropriate fields in the command dialog.
/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. If there is no existing value and you provide an optional *default* value, INPUT will display the default value for editing.
/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.

For example:

```
input /k"[0-9]-()" Enter your phone number: %%var
```

You can specify extended keys by enclosing their names in square brackets (within the quotes), See Keys and Key Names 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-RT will beep and wait for another keystroke.

/Lmax[:min]	Sets the maximum number of characters which INPUT will accept to ***max***. 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). The optional ***min*** parameter will set the minimum number of characters that INPUT will accept.
/N	Disables the use of input colors defined in the Colors configuration options, and forces INPUT to use the default display colors.
/P	Tells INPUT to echo asterisks, instead of the characters you type.
/T	Display a countdown timer (/W*n* is also required).
/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.

4.2.91 INSTALLED

Purpose: Display the apps installed on the system

Format: INSTALLED [/P[*n*] /A["*xxx*"] /D["*xxx*"] /I["*xxx*"] /U["*xxx*"] /V["*xxx*"] /X[86][64] *appname*]

      appname

      [/A (date)](#)   [/P(ause)](#)
      [/D(irectory)](#)   [/U (publisher)'](#)
      [/I(con)](#)   [/V(ersion)](#)

Usage:

Appname can contain wildcards. If you don't specify *appname*, it will default to *. The optional arguments after /A, /D, /I, /U, and /V will filter the results (they all can contain wildcards).

Example:

To show all of the installed x64 apps that have "1.0" somewhere in their version number:

```
installed /v"*1.0*" /x64
```

Option:

/A   Show the installed date (may be empty)

/D   Show the installation directory

/I   Show the icon file

/P[*n*]   Pause after displaying each page. Your options at the prompt are explained in detail under [Page and File Prompts](#). The /P option has an optional argument *n* that specifies the number of seconds to wait for a keystroke before continuing.

/U   Show the publisher

/V   Show the version number

4.2.92 INTERNAL

Purpose: Run an internal TCC command

Format: INTERNAL *command args ...*

Usage:

INTERNAL runs the specified TCC internal command, ignoring any aliases, plugins, functions, or external apps.

If *command* is not an internal TCC command, INTERNAL will return an error.

4.2.93 JABBER

Purpose: Send an IM via the JABBER network

Format: JABBER [/S"*server*" /U"*user*" / P"*password*" /IPv6 /Tn /V] /B *target*[@*server*] /F"*filename*" *message*

                message   The message to send

/B(uddy)	/S(erver)
/F(ile)	/Tn (port)
/IPv6	/U(sername)
/P(assword)	/V(erbose)

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.

The JABBER command supports SSL, so it can talk with SSL XMPP servers (like talk.google.com).

Options:

     /B   Address where the message will be sent

     /F   Send a file to another user

     /IPv6   Use IPv6 instead of IPv4

     /P   Logon password on the JABBER server

     /S   JABBER server to log onto

     /T   Server port (default 5222)

     /U   User logon name on the JABBER server

     /V   Display verbose (debugging) output

4.2.94 JAR

Purpose: Add, update, or delete files in a Java .JAR archive

Format: JAR [/A:[[-][+]rhsdaecjot] /A /C /D /F /L*n* /M /Ne /Nt /O:[-]adegnrstu /P /Q /R /TEST /U /V]
jararchive [@*file*] *file*...

> **jararchive**   The jar file to work with
> **file**   The files(s) to be added to the jar file

/A:... (attribute switch)   /O:... (sort order)
/A(dd)   /P(rogress)
/C(ontents)   /R(ecurse)
/D(elete)   /TEST
/F(reshen)   /U(pdate)
/M(ove)   /V(iew)

See also UNJAR.

Usage:

After compression, the .JAR file may then be imported into Java code or executed by a JVM. The syntax is similar to the ZIP command:

Options:

/A:...   Select only those files that have the specified attribute(s) set. See Attribute Switches for information on the attributes which can follow /A:. Do not use /A: with **@*file*** lists. See @file lists for details.

You can specify **/A:=** to display a dialog to help you set individual attributes.

/A   Add the specified file(s) to the Jar file. (This is the default.)

/C   Display (on standard output) the contents of a file in the Jar archive.

/D   Delete a file in the Jar archive.

/F   Update only those files that currently exist in the tar file, and which are older than the files on disk.

/L   Set the compression level (0-6)..

/M   Delete the files from the disk after adding them to the tar file.

/O:...   Sort the files before processing.

You may use any combination of the sorting options below. If multiple options are used, the files 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, instead of numerically when numeric substrings are included in the name or extension.
d	Sort by date and time (oldest first); also see **/T:acw**
e	Sort by extension
g	Group subdirectories first, then files
r	Reverse the sort order for all options
s	Sort by size
t	Same as **d**
u	Unsorted

/P   Display the progress (0 - 100%) for each file as it is archived.

/Q   Don't display the files being archived.

/R   If the argument is a subdirectory, copy all of the files in that subdirectory and all of its subdirectories to the tar archive.

/TEST   Test the integrity of the TAR file (header and contents). Any errors will be displayed on STDERR.

/U   Update files which either don't exist in the tar, or which are older than the files on disk.

/V   View the list of files in the tar file (date, time, size, and filename

4.2.95 JOBMONITOR

Purpose: Monitor Windows job activity

Format: JOBMONITOR [/C [*jobname*]]
JOBMONITOR *jobname* [* TIME PROCESS MEMORY] *n command*

      jobname **-** The Windows job to monitor

      * - Monitor all activity for the job

      TIME - Monitor the end of job and process time notifications

      PROCESS - Monitor the process notifications (process limit, new process, zero processes, end of process, abnormal process exit)

      MEMORY - Monitor the job and process memory limit notifications

      /C(lear)

See also JOBS.

Usage:

JOBMONITOR will set four environment variables when a condition is triggered:

    _jobaction - The type of notification, which will be one of these values:

        EndOfJobTime
        EndOfProcessTime

ActiveProcessLimit
ActiveProcessZero
NewProcess
ExitProcess
AbnormalExitProcess
ProcessMemoryLimit
JobMemoryLimit

_jobpid - The process PID

_jobprocessname - The name of the process

_jobtime - The time (hh:mm:ss.ms) the notification was received

Option:

/C   Delete the job monitor(s)

4.2.96   JOBS

Purpose: Create Windows Jobs and optionally attach processes to a job

Format: JOBS
[/J=*jobname* /N=*jobname* /B /C /D /G /K /R /S /U /W /X /Y /JM=*mem* /PM=*mem* /P=n /J T=*ms* /PT=*ms*] [*pid* | *processname*]

pid - Process ID of a process to assign to the job

processname - Process name of a process to assign to the job

/B(reakaway)	/R(ead clipboard)
/C(lose)	/S (SystemParametersInfo)
/D(esktops)	/U(ser handles)
/G(lobal atoms)	/W(rite clipboard)
/I(nfo)	/X (no logout or shutdown)
/K (close on last handle)	/Y (allow breakaway)
/L (display settings)	/Z (die on unhandled exception)

See also JOBMONITOR and START.

Usage:

A *job* in Windows allows you to control of one or more processes as a group. A job's basic function is to allow groups of processes to be managed as a unit. You can limit the amount of memory or cpu time for a job, and put restrictions on what processes in that job are allowed to do. A process can be a member of only one job object, and once a process is associated with a job, the association cannot be broken. After a process is associated with a job, by default any child processes it creates are also associated with the job. (See the /B option below for the exception to this rule.)

You can start a new job attached to a specific job with the "START /job=jobname" option.

You cannot attach a process to a job if that process already belongs to a job.

Examples:

Start a program, create a job named "NoStop", prevent the program (and any programs it starts) from logging out, rebooting, or shutting down, and terminate all of the processes when the last job handle is closed:

```
start /pgm myapp.exe
jobs /N=NoStop /X /K %_startpid
```

Option:

/J=*name* Set or display options for an existing job.

/N=*name* Create a new job.

/JM=*n* Causes all processes associated with the job to limit the job-wide sum of their committed memory to *n* bytes. When a process attempts to commit memory that would exceed the job-wide limit, it fails.

/PM=*n* Limit the maximum committed memory for for all processes in the job to *n* bytes. When a process attempts to commit memory that would exceed the per-process limit, it fails.

/P=*n* Limit the total number of processes in the job to *n.*

/JT=*ms* Limit the maximum amount of per-job user-mode execution time to *ms* milliseconds.

/PT=*ms* Limit the maximum amount of user-mode execution for all processes associated with the job to *ms* milliseconds.

/B If any process associated with the job creates a child process using the **CREATE_BREAKAWAY_FROM_JOB** flag while this limit is in effect, the child process is not associated with the job.

/C Close a job handle.

/D Prevent processes associated with the job from creating and/or switching to other desktops.

/G Prevent processes associated with the job from accessing global atoms.

/I Display limit info for the job

/K All processes associated with the job will terminate when the last handle to the job is closed.

/L Prevent processes associated with the job from calling the ChangeDisplaySettings API

/R Prevent processes associated with the job from reading from the Windows clipboard.

/S	Prevent processes associated with the job from changing system parameters using the SystemParametersInfo API.
/U	Prevent processes associated with the job from using USER handles owned by processes not associated with the same job.
/W	Prevent processes associated with the job from writing to the Windows clipboard.
/X	Prevent processes associated with the job from logging out of Windows, rebooting, or shutting down.
/Y	Allow any process associated with the job to create child processes that are not associated with the job.
/Z	Disables the critical error popup dialog for each process associated with the job. If an exception occurs, this will cause termination of the process with the exception code as the exit status.

4.2.97   JOINDOMAIN

Purpose: Join a computer to a domain or workgroup

Format: JOINDOMAIN [/W] *computer\domain*[*\organization*] *user* [*password*]

computer	The DNS or NETBIOS name of the computer.
domain	The name of the domain or workgroup to join.
organization	(Optional) The RFC 1779 format name of the organizational unit (OU) for the account. If you specify this parameter, it must contain a full path. (For example, OU=testOU,DC=domain,DC=Domain,DC=com.)
user	The account name to use when connecting to the domain controller. The name must be either a domain NetBIOS name and user account (for example, *jpsoft\rconn*) or the user principal name (UPN) of the user in the form of a login name (for example, "user@tcmd.com").
password	The password to use when connecting to the domain controller. If the password is not entered (or is *), TCC-RT will prompt for the password.

/W(orkgroup)

Usage:

Option:

/W	Join a workgroup instead of a domain

4.2.98   KEYBD

Purpose:	Set the state of the keyboard toggles Caps Lock, Num Lock, and Scroll Lock, or enable/disable the keyboard.
Format:	KEYBD [/C*n* /K[0|1] /N*n* /S*n*]

n can be either **0** to toggle the key off or **1** to toggle the key on.

/C(aps lock)   /N(um lock)
/K(eyboard lock)   /S(croll lock)

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.
/K	Disable (0) or enable (1) the keyboard. You can also reenable a disabled keyboard with Ctrl-Alt-End.
/N	Turn the Num Lock key on or off.
/S	Turn the Scroll Lock key on or off.

4.2.99   KEYS

Purpose:	Enable, disable, or display the history list
Format:	KEYS [ON	OFF	LIST]

Usage

This command is provided for compatibility with KEYS command in CMD, which controls the history list in Windows. (CMD*'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-RT will show you the current state of KEYS.

4.2.100 KEYSTACK

Purpose:	Send keystrokes to a program or command automatically
Format:	KEYSTACK [/I=*pid,ms* /I"*title",ms* /R *filename*] [/W*x*] ["*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.

/I(nput idle)   /W(ait)
/R(ead file)

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-RT itself), you must start the program or ACTIVATE 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 command, then use the KEYSTACK command. If you start the program directly (without using START) the batch file will wait for the application to complete before continuing and running the KEYSTACK command, and the keystrokes will not appear in the target program.

If you use KEYSTACK in an alias executed from the prompt, the considerations are essentially the same, but depend on whether or not the Wait for External Apps 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 is set, TCC-RT 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 /W option, and a repeat count. If you want to enter a double quote, use two double quotes. Do not prefix or append the two double quotes to a string argument.) For example, to insert the string *abc "def"*

```
keystack "abc " "" "def" ""
```

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 for a complete listing of key names and a description of the key name and numeric key code format.

Note

You may need to experiment with your programs and insert delays (see the /W option) to find the window activation and keystroke sequence that works for a particular program.

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 1/18 second each) for **Word** to get started, places the keystrokes for **Alt-F** (**File** 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 and KEYSTACK are issued on a single command line. This ensures that the keystrokes are sent to **Word**'s window, not back to TCC-RT.

Option:

/I   Wait for an input idle or the specified number of milliseconds.

/I=pid,milliseconds	Look for the specified process ID
/I"Title",milliseconds	Look for the specified window title

/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.

4.2.101 LIBRARY

Purpose: Create, modify, delete, or display library functions

Format: LIBRARY [/D func /F [func] /P[*n*] /Q /R file ... /U] [*command line*]

/D(elete)	/Q (no errors)
/F (display functions)	/R(ead functions)
/P(ause)	/U(pdate functions)

Usage:

LIBRARY will load / display / delete library functions, which are similar to batch files but which are loaded into RAM and can be called as if they are internal commands. Library functions are read from files, with the syntax:

```
functionname {
    command1
    command2
    ...
}
```

The opening brace **{** must be on the same line as the function name (separated by a space), and the closing brace **}** must be on a line by itself.

When TCC-RT starts, it will automatically load any library function files in the LIBRARY subdirectory of the TCC-RT installation directory. You can specify a different location by setting the LibraryDirectory INI directive. You can have any number of functions in a file.

You can prevent TCC-RT from loading library function files at startup with the TCC-RT /IL option.

If you do not specify any switches, LIBRARY will display the library function names that match the command line argument(s). If you do not specify any arguments, LIBRARY will display all of the library function names.

The command line following the library function name is passed to the function, and the arguments can be referenced with the same %1 - %*n* syntax as used by batch files and aliases.

Library functions can call aliases, internal or external commands, batch files, or other library functions.

You can specify which library to use for a function name (allowing you to use the same function names in different libraries). To specify a particular library and function, use the syntax:

library$function

Where *library* is the library file name, and *function* the name of the function.

If you don't specify a library name, TCC-RT will use the first matching function name it finds in the library list.

The TCC-RT parser will look for a matching library function name before looking for plugins, internal commands, external commands, or batch files. The command line following the library function name is passed to the function, and the arguments can be referenced with the same %1 - %n syntax as used by batch files and aliases.

The TCC-RT startup option **/IL** will prevent loading the default library functions (from the Library folder).

Prefixing a command name with a * will ignore alias and library function name matches.

Options:

/D	Delete a function (the function name can contain wildcards)
/F	Display the loaded (matching) functions (the function name can contain wildcards)
/P[*n*]	Pause after each page when displaying functions. Your options at the prompt are explained in detail under Page and File Prompts. The /P option has an optional argument *n* that specifies the number of seconds to wait for a keystroke before continuing.
/Q	Don't display an error if deleting or updating functions that don't exist
/R	Read function files. You can use multiple files with one LIBRARY /R command by placing the names on the command line, separated by spaces:

```
library /r function1.lst function2.lst
```

/U	Update function (otherwise you will get an error when loading a function that already exists)

4.2.102 LINKS

Purpose: Display the hardlinks for the specified file(s)

Format: LINKS [@*file*] *file*...

@file   A text file containing the filenames, one per line (see @file lists for details).

File Selection

Supports multiple file names.

Example:

```
[d:\temp] echo foo > foo
```

```
[d:\temp] links foo
D:\temp\foo

[d:\temp] md temp2

[d:\temp] mklink /h temp2\bar foo

[d:\temp] links foo
D:\temp\temp2\bar
D:\temp\foo
```

4.2.103 LIST

Purpose	Display a text file, with forward and backward paging and scrolling

Format	LIST [*range*...] [/8 /A:[[-|+]rhsadecijopt /B[-]*n* /C /Etext" /F /H /I /L[-]*n* /N /O:[-] adegnrstu /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 for details).
range	A file selection range (date, description, exclusion, size, time)

/8 (UTF-8)	
/A: (Attribute select)	/N (line numbers)
/B(yte offset)	/O:... (Order)
/C (separate console)	/R(everse)
/E (regular expression)	/S(tandard input)
/F (console screen buffer)	/T (search for Text)
/H(igh bit off)	/U (Ruler)
/I(gnore wildcards)	/W(rap)
/L(ine offset)	/X (heXadecimal display mode)

See also: HEAD, TAIL, and TYPE.

File Selection

Supports attribute switches, extended wildcards, ranges, multiple file names, and include lists.

Internet

Can be used with FTP/HTTP Servers.

Usage

LIST provides a fast and flexible way to view a file, without the overhead of loading and using a text editor.

For example, to display a file called *MEMO.DOC*:

```
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 32,767 characters will be truncated unless you're in Wrap or Hex modes.

LIST displays files in the TCC-RT window. If you resize the TCC-RT window or the *Take Command* window when *TCC* is running in a tab window, LIST will automatically resize its display.

You can define all of the LIST keys using the OPTION / Keyboard dialog.

Directive	Default	Description
ListBack	B	Return to the previous file.
ListClipboard	Ctrl-B	Copy the current filename to the clipboard.
ListContinue	C	Continue with the next file.
ListDelete	Del	Delete the current file.
ListDown	Down	Scroll down one row.
ListEdit	E	Edit the file with the editor associated with that filetype. If there is no association, LIST will use the editor defined in the Editor configuration option. If no editor is defined, LIST will use Notepad. 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.)
ListEnd	End	Go to the end of the file.
ListExit	Esc	Exit the current file.
ListFind	F	Prompt and search for a string or a sequence of hexadecimal values.
ListFindPrevious	Ctrl-N	Find previous matching string in the file.
ListFindRegex	R	Prompt and search for a regular expression.
ListFindRegexReverse	Ctrl-R	Search backwards for a regular expression.
ListFindReverse	Ctrl-F	Prompt and search for a string, searching backward from the end of the file.
ListGoto	G	Display a dialog to jump to a specific line.
ListHelp	F1	Display help for LIST.
ListHex	X	Toggle the hex mode (/X) option.
ListHexSpace	S	Toggle display of nonprintable characters (space or .) when in hex mode.
ListHighBit	H	Toggle the "strip high bit" (/H) option.
ListHome	Home	Go to the beginning of the file.
ListInfo	I	Displays information about the current file.
ListLeft	Left	Scroll left one column.
ListNext	N	Find next matching string.
ListNumber	L	Number the lines.
ListOpen	O	Display the "Open File" dialog.
ListPageLeft	Ctrl-Left	Scroll left 40 columns.
ListPageRight	Ctrl-Right	Scroll right 40 columns.
ListPgUp	PgUp	Scroll up one page.
ListPgDn	PgDn	Scroll down one page.
ListPrint	P	Print all or part of the file (displays the Windows "Print" dialog).
ListRefresh	F5	Refresh the display.
ListRight	Right	Scroll right one column.
ListSave	Ins	Save to a file.

ListTabSize	Tab	Display a dialog to set the tab size.
ListUnicode	U	Toggle the Unicode display mode.
ListUp	Up	Scroll up one row.
ListWrap	W	Toggle the "line wrap" (/W) option.

Text searches performed with **F**, **N**, **Ctrl-F**, and **Ctrl-N** 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 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 in the search string. For example, you can search for the string *to*day* to find the next line which contains the word *to* followed by the word *day* later on the same line, or search for the numbers *101* or *401* with the search string **[14]01**. If you begin the search string with a back-quote `` ` ``, or enclose it in back-quotes, wildcard characters in the string will be treated as normal text with no special wildcard meaning.

You can use the /T switch to specify search text for the first file. When you do so, LIST begins a search as soon as the file is loaded. Use /I to ignore wildcards in the initial search string, and /R to make the initial search go backwards from the end of the file. When you LIST multiple files with a single LIST command, these switches affect only the first file; they are ignored for the second and subsequent files.

You can also search using Regular Expressions using the **R** and **Ctrl-R** keys. See Regular Expression Syntax 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 32,767 characters, whichever comes first.

LIST normally allows long lines in the file to extend past the right edge of the screen. You can use the horizontal scrolling keys (see above) to view text that extends beyond the screen width. If you use the W command or /W switch to wrap the display, each line is wrapped when it reaches the right edge of the screen, and the horizontal scrolling keys are disabled.

To view output from another command simply pipe the output of the command to LIST, for example:

```
dir | list
```

Normally LIST will detect input from a pipe automatically, but if it does not, use /S to explicitly specify piped input. Your ability to navigate backward through the displayed output (*e.g.* with **PgUp**) may be

limited when viewing a very 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 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 as well as the contents of HTTP/HTTPS URLs. For example:

```
list ftp://ftp.microsoft.com/index
list https://jpsoft.com/notfound.htm
```

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 and IFTP.

- **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 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 is not redirected), LIST will open the common Windows "open file" dialog.

Most of the LIST keystrokes can be reassigned with key mapping directives.

By default, LIST sets tab stops every 8 columns. You can change this behavior with the Tabs Width configuration option.

Options

    /8   The file is interpreted as UTF-8.

/A: Select only those files that have the specified attribute(s) set. See Attribute Switches for information on the attributes which can follow /A:. Do not use **/A:** with **@file** lists. See @file lists for details.

You can specify **/A:=** to display a dialog to help you set individual attributes.

/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. /C only works in stand-alone TCC-RT windows, not in **Take Command** tab windows.

/E   Search for a regular expression 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**.

/F   Display the contents of the console screen buffer.

/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 "text" option. Directs LIST to interpret characters such as *, ?, [, and] as literal characters instead of wildcard characters. **/I** affects only the initial search started by /T, not subsequent searches started from within LIST.

/I"text"   Select files by matching text in their descriptions. See Description Ranges 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.

/O:...   Sort the files before processing.

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, instead of numerically when numeric substrings are included in the name or extension.
- **d**   Sort by date and time (oldest first); also see **/T:acw**
- **e**   Sort by extension
- **g**   Group subdirectories first, then files
- **r**   Reverse the sort order for all options
- **s**   Sort by size

t	Same as **d**
u	Unsorted

/R   Only meaningful when used in conjunction with the /T "***text***" option. Directs LIST to search for text from the end of the file instead of from the beginning of the file. Using this switch can speed up searches for text that is normally near the end of the file, such as a signature. /R affects only the initial search started by /T, not subsequent searches started from within LIST.

/S   Read from standard input rather than a file. This allows you to redirect command output and view it with LIST. Normally, LIST will detect input from a redirected command and adjust automatically. However, you may find circumstances when /S is required. For example, to use LIST to display the output of DIR you could use either of these commands:

```
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. 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 and /R.

/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.

4.2.104 LOADBTM

Purpose: Switch a batch file to or from BTM mode

Format: LOADBTM [ON | OFF]

Usage:

TCC-RT recognizes three kinds of batch files: *.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 a batch 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.

There is no functional difference between *.BAT* and *.CMD* files.

4.2.105 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.

4.2.106 LOCAL

Purpose: Define variables that are local to a library function or to a batch file.

Format: LOCAL *var1, var2, ...*

Usage:

LOCAL will save the existing values of the specified environment variables (if any) and then delete the variable from the environment. You can then SET a new variable with that name; when the library function or batch file exits, the local variables are deleted from the environment and the previous values (if any) are restored.

LOCAL allows you to use variables in your library functions without worrying about whether they are also used in the master environment or other library functions.

See also SETLOCAL.

Example:

This library function defines three local variables, which are only valid inside the function:

```
testfunc2 {
    local test, computer, server
    set test=abc
    set computer=Asus
    set server=PrintServer
    command1
    command2
    ...
}
```

4.2.107 LOCKMONITOR

Purpose: Monitor when the Windows session is locked or unlocked.

Format: LOCKMONITOR [/C [*action*]]
               LOCKMONITOR [Locked | Unlocked] [*n* | FOREVER] *command*

                n   Number of repetitions (or **FOREVER**)
                command   Command to execute when condition is triggered

                /C(lear)

Usage:

The command line will be parsed and expanded before LOCKMONITOR 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 the last argument on the line is a single **(,** it is interpreted as the beginning of a command group. LOCKMONITOR will append the following lines (in a batch file) or prompt you for more input (at the command line) until it gets a closing **).**

If you don't enter any arguments, LOCKMONITOR 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-RT thread is also performing I/O. You may need to use START or DETACH in **command** to avoid conflicts.

Options:

    /C   Remove the session lock monitor.

4.2.108 LOG

Purpose: Save a log of commands to a 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.
ON	Turns on logging
OFF	Turns off logging

/A(ll)	/H(istory log)
/E(rrors)	/W(rite to)

Usage:

The LOG command provides independent controls for four different methods of logging TCC-RT activity:

- Command Log
- Error Log
- History Log
- Output Log

If you don't specify a log type, LOG defaults to command logging. You can only specify one of the /A, /E, and /H options in a single LOG command.

You can have any combination of the four logging methods running simultaneously.

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, and ***command*** is the actual command after any alias or variable expansion.

The default command log filename is ***TCCommandLog***. See also %_LOGFILE.

Error Log

Error logging saves all error messages to the ***error log***. The default filename is ***TCErrorLog***.

History Log

History logging 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. See also %_HLOGFILE.

Output Log

The Output log saves everything that TCC-RT writes to the console window. It does not log output written by external applications.

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 *TCErrorLog*.

/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-RT session. Turning LOG or LOG /H off or on does not change the file name.

4.2.109 LUA

Purpose: Invoke the internal Lua interpreter

Format: LUA [*options*] [*script* [*args*]]

-e *_stat_*	executes string stat;
-l *_mod_*	"requires" mod;
-i	enters interactive mode after running script;
-v	prints version information;
--	stops handling options;
-	executes stdin as a file and stops handling options.

Usage:

The internal Lua is version 5.4.4.

After handling its options, lua runs the given script, passing to it the given args as string arguments. When called without arguments, lua behaves as lua -v -i when the standard input (stdin) is the console, and as lua - otherwise.

Before running any argument, the interpreter checks for an environment variable LUA_INIT. If its format is @_filename_, then lua executes the file. Otherwise, lua executes the string itself.

All options are handled in order, except -i. For instance, an invocation like

```
lua -e "a=1" -e print(a) script.lua
```

will first set **a** to 1, then print the value of **a**, and finally run the file script.lua with no arguments.

Before starting to run the script, lua collects all arguments in the command line in a global table called arg. The script name is stored at index 0, the first argument after the script name goes to index 1, and so on. Any arguments before the script name (that is, the interpreter name plus the options) go to negative indices. For instance, in the call

```
lua -la b.lua t1 t2
```

the interpreter first runs the file a.lua, then creates a table

```
arg = {
[-2] = "lua",
[-1] = "-la",
[0] = "b.lua",
[1] = "t1", [2] = "t2" }
```

and finally runs the file b.lua. The script is called with arg[1], arg[2], ... as arguments; it can also access these arguments with the vararg expression '=...='.

In interactive mode, if you write an incomplete statement, the interpreter waits for its completion by issuing a different prompt.

If the global variable _PROMPT contains a string, then its value is used as the prompt. Similarly, if the global variable _PROMPT2 contains a string, its value is used as the secondary prompt (issued during incomplete statements). Therefore, both prompts can be changed directly on the command line. For instance,

```
lua -e"_PROMPT='myprompt> '" -i
```

(the outer pair of quotes is for the shell, the inner pair is for Lua), or in any Lua programs by assigning to _PROMPT. Note the use of -i to enter interactive mode; otherwise, the program would just end silently right after the assignment to _PROMPT.

4.2.110 MD / MKDIR

Purpose: Create a subdirectory

Format: MD [/C /D /N[et] /S] *path*...
             or
        MKDIR [/C /D /N[et] /S] *path*...

        path   The name of one or more directories to create.

        /C(ompressed)   /N(o update)
        /D (change directory)   /S(ubdirectories)

See also: RD.

Internet: Can be used with FTP Servers.

Usage:

MD and MKDIR are synonyms. You can use either one.

MD creates a subdirectory anywhere in the directory tree. To create a subdirectory from the root, start the **path** with a backslash [\]. For example, this command creates a subdirectory called *MYDIR* in the root directory:

```
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 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 database unless the **/N** option is specified.

You can create directories on FTP servers. For example:

```
md ftp://ftp.abc.com/data/index
```

MD sets two internal variables:

%_md_dirs	The number of directories created
%_md_errors	The number of errors

Options:

/C   Create a compressed subdirectory.

/D   Change to the newly created subdirectory.

/N   If **/N** has no additional options, do not update the CD / CDD extended directory search 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 non-fatal errors. (Note that a **/Ne** alone will still update the extended directory search database.)
  t   Don't update the extended directory search 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, **/S** becomes the default if you enable TCC-RT extensions with the **/X** switch on the **TCC**startup command line. See Command Line Options for details on **/X**.

4.2.111 MEMORY

Purpose: Display TCC-RT and Windows memory status

Format: MEMORY

Usage:

MEMORY lists the percentage "memory load" as reported by Windows, the total and available physical RAM, the total and available page file size, the total and available virtual memory, the total and free alias space (local and/or global), the total and free function space (local and/or global), the total history space, the current and maximum working set for TCC-RT, and the private memory usage for TCC-RT. The memory load is a figure returned by the operating system which gives an overall sense of memory utilization. It is not a precise indicator of system load or memory usage. The total page file figure shows the total number of bytes that can be stored in the file, but may not reflect the actual size of the current file on disk.

4.2.112 MKLINK

Purpose: Create NTFS symbolic, hard, and soft links

Format: MKLINK [/A:[[-]rhsadecijopt /A /D /H /J /Q /X] *Link Target*

Link	The new symbolic link name
Target	The pathname (full or relative) that the new link refers to

/A	Create a link with an absolute path.
/D	Create a directory symbolic link. (The default is to create a file symbolic link.)
/H	Create a hard link (like MKLNK).
/J	Create a junction.
/Q	Don't display results.
/X	Delete directory link.

File Selection

Supports attribute switches, extended wildcards, ranges, and include lists. 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 for details.

Usage:

If you don't specify a target or any options, MKLINK will display information on the link (including OpenAFS reparse points).

Due to Windows file system restrictions, creating symbolic links with MKLINK requires an NTFS volume.

The file/directory names in **Link** and **Target** can be fully or partially qualified. MKLINK will also copy an existing description to the link.

MKLINK sets two internal variables:

%_mklink_files   The number of links created
%_mklink_errors   The number of errors

See also MKLNK.

Example:

Create a symbolic file link "c:\mydir\myfile" that refers to the existing file "c:\data\somefile" :

```
mklink c:\mydir\myfile c:\data\somefile
```

Option:

/A   Create a link with an absolute (full expanded) pathname. For CMD compatibility, MKLINK creates relative links if you don't specify a full pathname.

/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.

/X   Delete a directory link.

4.2.113 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) or directory (for soft link).
              parm2   Name of the new directory entry (a file or directory reference) to be created.

              /A: (Attribute select)
              /D   Delete a link

              See also MKLINK.

File Selection

For hard links, MKLNK supports attribute switches, extended wildcards, ranges, multiple file names, and include lists. 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 for details.

Usage:

MKLNK is obsolete; you should use MKLINK for new scripts.

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.

MKLNK sets two internal variables:

%_mklnk_files	The number of links created
%_mklnk_errors	The number of errors

Hard Links

If **parm1** is a file, and **parm2** does not exist, MKLNK will create a hard link. 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, also known as a "directory junction" or "reparse point". If **parm2** exists, and it is a soft link, 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](#) 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.

4.2.114 MONITOR

Purpose:	Display or change monitor capabilities
Format:	MONITOR [/AP:*x:y* /AS:*x:y* /B:n /C:n /D:*Color:n* /FC /FD /G:*Color:n* /N:*n* /S /T:n]

[/AP:x:y (display position)](#)	[/FC (factory colors)](#)
[/AS:x:y (display area)](#)	[/FD (factory defaults)](#)
[/B:n (brightness)](#)	[/N:n (physical monitor)](#)
[/C:n (contrast)](#)	[/S (save settings)](#)
[/D:color:n (drive value)](#)	[/T:n (color temperature)](#)
[/G:color:n (gain value)](#)	

Usage:

MONITOR can display or change monitor capabilities, including:

> Technology type
> Color temperature
> Contrast
> Display area position
> Display area size
> RGB drive
> RGB gain
> Brightness
> Reset factory color defaults
> Reset factory defaults
> Save to nonvolatile storage

Not all settings are supported by all monitors. If you don't enter any arguments, MONITOR will display the current configuration of all physical monitors. Depending on the options and the monitor hardware, MONITOR can take several hundred milliseconds to return.

The MONITOR command will fail if the monitor does not support DDC/CI.

Example:

Set the second monitor to a brightness of 90 and a contrast of 75:

```
monitor /n:1 /b:90 /c:75
```

Options:

/AP:*x:y*	Set the horizontal (x=0) or vertical (x=1) position of the monitor's display area. *y* is the new width or height. Increasing the horizontal position moves the display area to the right; decreasing it moves the display area to the left. Increasing the vertical position moves the display area up, decreasing it moves the display area down.
/AS:*x:y*	Set the display area width (x=0) or height (x=1). *y* is the new width or height.
/B:*n*	Set the brightness.
/C:*n*	Set the contrast.
/D:*Color:n*	Sets a monitor's red, green, or blue drive value. Drive settings are used to adjust the monitor's white point (*drive* is also called *black level*). *Color* is either RED, GREEN, or BLUE; *n* is the drive value (usually 0-100). You can have multiple /Drive arguments in a single MONITOR command.
/G:*Color:n*	Sets a monitor's red, green, or blue gain value. Gain settings are generally used to adjust the monitor's white point. *Color* is either RED, GREEN, or BLUE; *n* is the gain value (usually 0-100). Changing the gain settings can change the color temperature. You can have multiple /Gain arguments in a single MONITOR command.
/FC	Restore the factory color settings.

/FD	Restore the factory default settings
/N:*n*	Change settings on physical monitor *n*. The default is 0.
/S	Save settings to the display's nonvolatile storage.
/T:*n*	Change the color temperature. *n* can be one of the following:

<div style="text-align:center">
4000
5000
6500
7500
8200
9300
10000
11500
</div>

4.2.115 MOUNTISO

Purpose: Mount an ISO image

Format: MOUNTISO [*d:* | *d:\path*] *image*

d:	Optional drive letter.
d:\path	Optional mount path
image	ISO file to mount

See also UNMOUNTISO.

Usage:

MOUNTISO is only supported in Windows 8 or later.

If you do not specify a drive letter or mount path, Windows will assign a drive letter.

You must be running an elevated session to mount an ISO image.

Example:

Mount the ISO image file "windows12.iso" as drive M:

```
mountiso m:\ windows12.iso
```

4.2.116 MOUNTVHD

Purpose: Mount a VHD or VHDX image

Format: MOUNTVHD [*d:* | *d:\path*] *image*

d:	Optional drive letter.
d:\path	Optional mount path
image	VHD or VHDX file to mount

See also [UNMOUNTVHD](#).

Usage:

If you do not specify a drive letter or mount path, Windows will assign a drive letter.

You must be running an elevated session to mount a VHD or VHDX image.

Example:

Mount the VHD image file "windows12.vhd" as drive M:

```
mountiso m:\ windows12.vhd
```

4.2.117 MOVE

Purpose:	Move files to a new directory (and optionally drive)
Format:	MOVE [/= /A:[[-]rhsadecijopt /B /C /CF /D /DD /DS:[acwu]yyyy-mm-dd /E /G /H /I"text" /J /K /L /LD /M /MD /MDA /N[dejnstz] /O /O:[-]acdeginorstuz /P /Q /R /S[[+]*n*] /SX /T /TS[acwu]hh:mm:ss.ms /U /UF /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](#) for details).

[/A: (Attribute select)](#)	[/MDA (Copy directory attributes)](#)
[/B (Move after reboot)](#)	[/N (Disable)](#)
[/C(hanged)](#)	[/O (don't move if target exists)](#)
[/CF (changed 2s+ resolution)](#)	[/O:... (Order)](#)
[/D(irectory)](#)	[/P(rompt)](#)
[/DD (delete empty subdirectories)](#)	[/Q(uiet)](#)
[/DS (date stamp)](#)	[/R(eplace)](#)
[/E (No error messages)](#)	[/S(ubdirectory tree)](#)
[/G (display percent copied)](#)	[/SX (single target directory)](#)
[/H(idden and system)](#)	[/T(otal)](#)
[/I"text" (match description)](#)	[/TS (time stamp)](#)
[/J (copy in restartable mode)](#)	[/U(pdate)](#)
[/K (delete to recycle bin)](#)	[/UF (updated 2s+ resolution)](#)
[/L (ASCII FTP transfer)](#)	[/V(erify)](#)
[/LD (create link)](#)	[/W(ipe)](#)
[/M(odified files)](#)	[/Y (force move of encrypted files)](#)
[/MD (Create target directory)](#)	[/Z (overwrite)](#)

See also [COPY](#), [DEL](#) and [RENAME](#).

File Selection

Supports <u>attribute switches</u>, extended <u>wildcards</u>, <u>ranges</u>, <u>multiple file names</u>, <u>delayed variable expansion</u>, and <u>include lists</u>. Date, time, size, or file exclusion ranges anywhere on the line apply to all *source* files. Use wildcards with caution on LFN volumes; see <u>LFN File Searches</u> for details.

Internet: Can be used with <u>FTP/TFTP/HTTP/HTTPS Servers</u>.

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 <u>COPY</u> and <u>RENAME</u>, MOVE works with single files, multiple files, and sets of files specified with an include list. If you don't specify any arguments, MOVE will display its command dialog.

The simplest MOVE command moves a single ***source*** file to a new location and, optionally, gives it a new name. These two examples both move one file from drive *C:* to the root directory on drive *A:*

```
[c:\] move myfile.dat a:\
[c:\] move myfile.dat a:\savefile.dat
```

In both cases, *MYFILE.DAT* is removed from drive *C:* after it has been copied to drive *A:*. If a file called *MYFILE.DAT* in the first example, or *SAVEFILE.DAT* in the second example, already existed on drive *A:*, it would be overwritten. (This demonstrates the difference between MOVE and RENAME. MOVE will move files between drives and will overwrite the destination file if it exists; RENAME will not.)

When you move a single file, the ***destination*** can be a directory name or a file name. If it is a directory name, and you add a backslash [\] to the end of the name, MOVE will display an error message if the name does not refer to an existing directory. You can use this feature to keep MOVE from treating a mistyped ***destination*** directory name as a file name, and attempting to move the ***source*** file to that name.

If you MOVE multiple files, the ***destination*** must be a directory name. MOVE will move each file into the ***destination*** directory with its original name. If the ***destination*** is not a directory, MOVE will display an error message and exit. For example, if *C:\FINANCE\MYFILES* is not a directory, this command will display an error; otherwise, the files will be moved to that directory:

```
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.

Be careful when you use MOVE with the <u>SELECT</u> 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.)

If you specify the /C, /CF, /R, /U, or /UF options, MOVE will append a **!** to the move specifier if the target exists and is being overwritten. For example:

```
[d:\] move file1 file2
file1 ->! file2
```

MOVE sets three internal variables:

%_move_dirs	The number of directories created
%_move_files	The number of files moved
%_move_errors	The number of errors

- **FTP Usage:**

You can move files to and from Internet URLs (FTP, TFTP and HTTP). For example:

```
move ftp://ftp.abc.com/f1.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 for additional details.

- **Advanced Features and Options**

If MOVE must physically copy the files and delete the originals (rather than renaming them), 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.

Use caution with the **/A:** and **/H** switches (both of which can allow MOVE to process hidden files) when you are physically moving files, and both the *source* and *destination* directories contain file descriptions. If the *source* file specification matches the description file name (normally *DESCRIPT.ION*), and you tell MOVE to process hidden files, the *DESCRIPT.ION* file itself will be moved, overwriting any existing file descriptions in the *destination* directory. For example, if the *C:\DATA* directory contains file descriptions, this command would overwrite any existing descriptions in the *D:\SAVE* directory:

```
[c:\data] move /h d* d:\save\
```

(If you remove the hidden attribute from the *DESCRIPT.ION* file the same caution applies even if you do not use **/A:** or **/H**, as *DESCRIPT.ION* is then treated like any other file.)

Note: The wildcard expansion process will attempt to allow both CMD-style "extension" matching (only one extension, at the end of the word) and the advanced *TCC string* matching (allowing things like *.*.abc) when an asterisk is encountered in the *destination* of a MOVE command.

MOVE supports [regular expression](#) back references in the target name. If you are using back references, you must also use a regular expression in the source name. The syntax is:

```
move ::filename ::target
```

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
```

You can override the default HTTP proxy server, proxy user, and proxy password (set in TCMD.INI) with the **/Proxy...** options.

/Proxy=*server*
/ProxyUser=*username*
/ProxyPwd=*password*

Options:

/=   Display the MOVE command dialog to help you set the filename and command line options. The /= option can be anywhere on the line; additional options will set the appropriate fields in the command dialog.

/A: Select only those files that have the specified attribute(s) set. See [Attribute Switches](#) 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](#) for details.

You can specify **/A:=** to display a dialog to help you set individual attributes.

/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 for details.

/CF   Move files only if the *destination* file exists and is more than 2 seconds older than the *source* (see also /U and **/UF**). Do not use /CF with @file lists. See @file lists 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.

/DD   When used with **/S**, MOVE will delete any empty source subdirectories.

/DS   Change the date timestamp on the target file(s) to the specified 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 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. /G will also display the % moved even if Windows is doing a rename (which may be a copy & delete internally).

/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 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.

/K   If the MOVE is to a different drive, move the source file to the recycle bin instead of deleting it. When deleting to the recycle bin, MOVE checks the RECYCLEEXCLUDE environment variable. If the file matches, MOVE deletes the file instead of sending it to the recycle bin.

/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 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!)

/MDA   Copy the attributes from the source subdirectories to the target subdirectories. Only valid if moving to another drive; otherwise MOVE does a rename of the top-level directory, and all of the subdirectory attributes are retained.

/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 or more 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)
n	Don't update the file descriptions
s	Don't display the summary.
t	Don't update the CD / CDD extended directory search database *(JPSTREE.IDX).*
z	Skip system directories (when used with /S)

/O   Don't move the file(s) unless the target doesn't exist, i.e. do not overwrite an existing target..

/O:...   Sort the files before processing.

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, instead of numerically when numeric substrings are included in the name or extension.
c	Sort by compression ratio
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 description
o	Sort by owner
r	Reverse the sort order for all options
s	Sort by size
t	Same as **d**
u	Unsorted
z	Same as **s**

/P   Prompt the user to confirm each move. Your options at the prompt are explained in detail under Prompts.

/Proxy=*server*

/ProxyUser=*username*

/ProxyPwd=*password*

/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. You cannot combine multiple /S options (including /S, /Sn, /S+1, or /SX) in a single command, or use any /S option with @file lists. See [@file lists](#) 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.

If you specify a **+** followed by a number after the /S, MOVE will not move any files until it gets to that depth in the subdirectory tree. For example, if you have a directory tree \a\b\c\d\e, /S+2 will not move anything in \a or \a\b.

MOVE will display the empty source subdirectories it is removing (unless you use the /Q option).

/SX Move the subdirectory tree to a single target directory (implies /S). MOVE will remove empty subdirectories after the move. You cannot combine multiple /S options (including /S, /Sn, /S+1, or /SX) in a single command, or use any /S option with @file lists. For example, to move all of the .EXE files in *c:\files* and all of its subdirectories to the directory *d:\exefiles*:

    copy /sx c:\files*.exe d:\exefiles\

/T Don't display filenames as they are moved, but display the total number of files moved.

/TS Change the time timestamp on the target file(s) to the specified time.

/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](#) 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.

/UF Move each *source* file only if it is more than 2 seconds newer than a matching *destination* file or if a matching *destination* file does not exist (also see **/C** and **/CF**). Do not use /UF with @file lists. See [@file lists](#) for details.

/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. /V will not work for FTP, TFTP, or HTTP moves.

/W   If the MOVE is to a different drive, after the move overwrite the source file contents using the DoD 5220.22-M (E) standard for secure deletion. (This overwrites every byte in the file three times with different values). 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.

/WAIT=_n_ Pause for _n_ milliseconds between each block copied from the source to the target file. This is useful for users with slow networks and very large file copies; it prevents MOVE from monopolizing all of the network I/O. (Only valid if the source and target are on different drives.)

/Y   Force copy of an encrypted file even when the target will be decrypted (for CMD 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.

4.2.118 MOVEDIR

Purpose: Move a directory tree

Format: MOVEDIR _source destination_

           source   The source directory tree
           destination   The target directory tree

Usage:

Both _source_ and _destination_ must be directory names. If _destination_ does not exist, MOVEDIR will create _destination_ and move _source_ to _destination_. If _destination_ already exists, MOVEDIR will append the last subdirectory name in _source_ to _destination,_ create the new subdirectory, and move _source_ to _destination_. (This allows you to rename the target directory.)

Examples

To move **_d:\test\mydir_** to **_x:\mydir:_**

```
movedir d:\test\mydir x:\
```

To move **_d:\test\mydir_** to **_x:\myolddir:_**

```
movedir d:\test\testmydir x:\myolddir
```

4.2.119 MSGBOX

Purpose: Display a Windows message box

Format: MSGBOX
[/1["*text*"] /2["*text*"] /3["*text*"] /4["*text*"] /5["*text*"] /6["*text*"] /B*rgb* /D
n /H /I /L /M /N /O /P*x,y* /Pc /Q /R /S /T*n* /W /X] *buttontype* ["*title*"] *prompt*

buttontype	One of **OK**, **OKCANCEL**, **YESNO**, **YESNOCANCEL**, **RETRYCANCEL**, **ABORTRETRYIGNORE**, **CANCELTRYCONTINUE**, **CONTINUEABORT**, **SKIPSKIPALLCANCEL**, or **IGNOREIGNOREALLCANCEL**
title	Text for the title bar of the message box.
prompt	Text that will appear inside the message box.

/1 (st button)	/M (system modal)
/2 (nd button)	/N (no sound)
/3 (rd button)	/O (topmost window)
/4 (th button)	/P (screen coordinates)
/5 (th button)	/Pc (center)
/6 (th button)	/Q(uestion icon)
/B (background color)	/R(ight justify buttons)
/F (foreground color)	/S(top icon)
/D(isable temporarily)	/T(imeout)
/H(elp button)	/W(arning icon)
/I(nformation icon)	/X (not moveable)
/L(imit width)	

See also: INKEY, INPUT, QUERYBOX, and TASKDIALOG.

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-RT will automatically size and center the message box on the tab window (if ***TCC*** is running in a ***Take Command***), or centered on the screen (if TCC-RT is running in a console window). The message box has up to three response buttons (plus an optional Help button), depending on its type, as shown below.

buttontype	*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	
SKIPSKIPALLCANCEL	Skip	Skip All	Cancel
IGNOREIGNOREALLCANCEL	Ignore	Ignore All	Cancel

There are two button type modifiers (only valid when used immediately following a YESNO or YESNOCANCEL):

    YESTOALL - adds a "Yes to All" button (returns 30)
    NOTOALL - adds a "No to All" button (returns 31)

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 %_?. Be sure to save the return value in another variable or test it immediately; because the value of %_? changes with every internal command. The following list shows the value returned for each selection:

response	% ?
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
custom button 5	25
custom button 6	26

If you define custom buttons, the button type argument will be ignored.

There are three optional Checkbox types. (You can only choose one at a time.)

    DONOTASKAGAIN - add checkbox "Do not ask me again".
    DONOTTELLAGAIN - add checkbox "Do not tell me again"
    DONOTSHOWAGAIN - add checkbox "Do not show again"

If the checkbox is selected, MSGBOX will set the internal variable %_msgbox_checkbox to 1.

If there is an error in the MSGBOX command itself, %_? will be set as described in its documentation (see _?).

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.

You can copy the text in a MSGBOX window to the clipboard by entering Ctrl-C when the MSGBOX window has the keyboard focus.

MSGBOX creates a popup dialog box. If you prefer to retrieve input from the command line, see the INKEY and INPUT 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.

/5   If there is a text string following the option, set the custom text for the fifth button. Otherwise, set the fifth button as the default.

/6   If there is a text string following the option, set the custom text for the sixth button. Otherwise, set the sixth button as the default.

/B*rgb* Background color, as a hex number where Blue is the most significant 2 bytes, Green the middle two, and Red the least significant. For example, /BAA8866 will set Blue to AA, Green to 88, and Red to 66.

/D*n*   Disable the message box buttons for *n* seconds at startup.

/F*rgb* Text color, as a hex number where Blue is the most significant 2 bytes, Green the middle two, and Red the least significant. For example, /FAA8866 will set Blue to AA, Green to 88, and Red to 66.

/H   Display a help button.

/I   Display an icon consisting of a lower case "i" in a circle in the message box.

/L   Limit the maximum message box width to no more than 1/3 the screen width (unless the button text requires more).

/M   The message box window will be displayed on top of all other windows.

/N   Don't play the default sound.

/O   The message box is created as a topmost window.

/P*x,y* The initial x,y screen coordinates. If you don't use this option, MSGBOX will center its window in the *Take Command* tab window or the TCC-RT console window.

/Pc   Center the MSGBOX window on the desktop.

/Q   Display a question mark icon in the message box.

/R   The buttons will be right-justified.

/S   Display a stop sign icon in the message box.

/T*n*   MSGBOX will wait a maximum of *n* seconds for a response (and then close). 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.

/X   The message box cannot be moved.

4.2.120 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)

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 the last argument on the line is a single **(,** it is interpreted as the beginning of a command group. NETMONITOR will append the following lines (in a batch file) or prompt you for more input (at the command line) until it gets a closing **).**

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-RT thread is also performing I/O. You may need to use START or DETACH 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.

4.2.121 ODBC

Purpose: Query a database through an ODBC driver

Format: ODBC [/O "*connectionstring*"]["*Query*"][/C]

/O - Send connection string
/C - Close

"*query*" - SQL query to executes

Usage:

Options:

/C Close the ODBC session.

/O Send the specified connection string to the ODBC driver. This opens a persistent ODBC session.

4.2.122 ON

Purpose: Execute a command in a batch file when a specific condition occurs

Format: ON BREAK [*command*]
ON CLOSE [*command*]
ON CONDITION [*condition command*]
ON DBLCLICK [*command*]
ON ERROR [*command*]
ON ERRORLEVEL *n* [*command*]
ON ERRORMSG [*command*]
ON LOGOFF [*command*]
ON LBUTTON [*command*]
ON MBUTTON [*command*]
ON RBUTTON [*command*]
ON RESUME [*command*]
ON SHUTDOWN [*command*]
ON SUSPEND [*command*]

command command to execute when the event occurs

Usage:

ON sets a watch that remains in effect for the duration of the current session or batch file, or until replaced by another ON command of the same type. Whenever a **break** or **error** condition occurs after ON has been executed, the corresponding **command** is automatically executed. You can have multiple ON commands active at a time, as long as no two are the same type. (For example, you can have an ON BREAK and an ON CLOSE, but not two ON LBUTTON.)

If the last argument on the line is a single **(,** it is interpreted as the beginning of a command group. ON will append the following lines (in a batch file) or prompt you for more input (at the command line) until it gets a closing **).**

Global Conditions:

The following ON conditions can be run from the command prompt (a "global condition"); all others will only work in a batch file.

    ON CLOSE
    ON LOGOFF
    ON SHUTDOWN
    ON SUSPEND
    ON RESUME

If no command is specified, TCC-RT will remove the existing command for the specified condition. Each time an ON statement is defined, it defines a new command to be executed for that event, and any prior command is discarded. If an ON condition is defined for the current batch file, it will override a global ON condition.

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 the TCC-RT session is closed.

Activation of **ON CONDITION**

ON CONDITION will execute ***command*** when ***condition*** is true. ***condition*** can be any test that is valid in [IF](). The test will be done after each command is executed. If you are executing a loop ([DO]() or [FOR]()), the test will be done each time through the loop.

Activation of **ON DBLCLICK**

ON DBLCLICK will execute ***command*** when the left mouse button is double clicked when TCC-RT is the active window. (Note that if you also have an ON LBUTTON command, it will be executed on the first click.)

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]() 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-RT error message (Windows errors may still be displayed). ON ERROR will also set the %_SYSERR internal variable.

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 equal to the integer specified by *n.* You can also use the IF ERRORLEVEL tests; for example:

```
ON ERRORLEVEL EQ 37 ...
```

Activation of **ON LBUTTON**

ON LBUTTON will execute *command* when the left mouse button is clicked.

Activation of **ON LOGOFF**

ON LOGOFF will execute *command* when the user logs off.

Activation of **ON MBUTTON**

ON MBUTTON will execute *command* when the middle mouse button is clicked when TCC-RT is the active window.

Activation of **ON RBUTTON**

ON RBUTTON will execute *command* when the right mouse button is clicked when TCC-RT is the active window.

Activation of **ON RESUME**

ON RESUME will execute *command* when the system resumes after sleeping or hibernating.

Activation of **ON SHUTDOWN**

ON SHUTDOWN will execute *command* when the system is being shut down.

Activation of **ON SUSPEND**

ON SUSPEND will execute *command* when the system is going to sleep or hibernation. Windows will continue suspending after a maximum of 2 seconds.

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.

If you do not specify a command, TCC-RT restores the 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 to another batch file) or permanently, the *TCC-RT* default **break** and **error** handlers become effective. A CALLed 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 are reactivated.

Limitations

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.

Examples:

The command can be any command that can be used on a batch file line by itself. Frequently, it is a GOTO or GOSUB command. For example, the following fragment traps any user attempt to end the batch file by pressing **Ctrl-C** or **Ctrl-Break**. It scolds the user for trying to end the batch file 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 as the command if you want to execute multiple commands, for example:

```
on break (echo Oops, got a break! & quit)
```

ON assumes 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,
- end the batch file with QUIT or CANCEL
- chain to another batch file (without using CALL).

When handling an error condition with ON ERROR[MSG], you may find it useful to use internal variables, particularly %_? and %_SYSERR, to help determine the cause of the error.

To force TCC-RT to ignore break or error, use the REM command as your command.

Options:

/G   Set a global condition (one that will be executed whether *TCC-RT* is in a batch file or at the command prompt). This is useful when you want to set global conditions from a batch file. For example:

```
ON /G LOGOFF command
```

4.2.123 OPTION

Purpose: Modify or display TCC-RT configuration

Formats: Check for updates:
OPTION /U

Temporarily changing an option:
OPTION *//directive=value* ...

Temporarily changing a list of options:
OPTION *@filename*

Displaying the current value of an option:
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, SETDOS

Usage:

Check for Updates

The /U option will invoke the updater to check https://jpsoft.com for updates to **TCC-RT**.

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. If you do not specify a value, OPTION will reset the value for that directive to the default.

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. 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 function.

4.2.124 OSD

Purpose: Write floating text to the display

Format: OSD
[/ID=
n /C[=
n] /Font=
n /ID=
n /N /POS=*top,left* /RGB=*r,g,b* /TIME=
n /TOP /BOTTOM /LEFT /RIGHT /HCENTER /VCENTER /V] *text*

/ID=n	Open the OSD window *n* (0-9). /ID is optional; it will default to 0. If /ID is specified, it must be the first argument.
/C=n	Close the specified OSD display. /C=n must be the only argument. /C will default to OSD window 0.
/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
text	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***.

You can control up to 10 simultaneous OSD windows with the /ID=n and /C=n options. If you don't specify /ID, OSD will default to window 0.

4.2.125 PATH

Purpose: Display or alter the list of directories that TCC-RT will search for executable files, batch files, and files with executable extensions that are not in the current directory

Format: PATH [/D *directory* /M /N /V] [*directory* [;*directory*...]]

directory   The full name of a directory to include in the path setting.

/D(elete)
/M(aster environment)
/N(ew line)
/V(erify directories)

See also: ESET and SET (the PATH command is syntactically equivalent to SET PATH).

Usage:

When TCC-RT is asked to execute an external command (an *.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-RT first searches the current directory before any directories listed in your search path. For example, after the following PATH command, TCC-RT will search for an executable file in four directories: the current directory, the root directory on drive C, then the *BIN* subdirectory on C, and then the *UTIL* subdirectory on C:

```
path c:\;c:\bin;c:\util
```

The list of **directories** to search is stored as an environment string, and can also be set or viewed with SET, and edited with ESET.

The PATHEXT environment variable 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.CMD).

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-RT 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.

Options:

    /D   Remove the specified directory from the PATH variable.

    /M   Reset the PATH variable to the original value when TCC-RT was started.

    /N   Display each PATH directory on its own line.

    /V   Checks all of the directories in %PATH, and displays an error message for any that don't exist.

4.2.126 PAUSE

    Purpose: Suspend batch file or alias execution

    Format: PAUSE [/W*n* /C /T] [*text*]

          /W*n*   Wait
          /C   Clear the prompt
          /T   Countdown timer
          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.

PAUSE will remove any spaces before the prompt text. If you want to indent the message, you can use back quotes to preserve spaces:

```
pause `      `Press Ctrl-C to abort ...
```

Options:

/C   After you press a key, erase the prompt and do not print a CR/LF.

/T   Displays a countdown timer. Must be used with /W*n*, which must be the first argument on the command line.

/W   Wait for a maximum of *n* seconds and then continue with the next command. If you combine /W and /C, /W must be the first argument on the command line.

4.2.127 PDIR

Purpose: Display information about files and subdirectories in user-definable fields. It is a "programmable DIR" command.

Format: PDIR [*ranges*] [/\ /A:[*attrlist*] /B CD:*text* /D /H /HL /I"*text*" /K /M /N[defhjlsvz] /O: [*order*] /P[*n*] /Q /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	/M	show footer
/B	Bare filenames	/N	Disable options
/CD:...	COLORDIR string	/O	Order
/D	colorize	/P	Page pause
/H	do not Hide . and ..	/Q	Owner name
/HL	Hard links	/S	Subdirectories
/I"text"	description range	/T[:t]	Timestamp type
/K	show header	/(...)	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 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 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)
       q   Enclose the filename in double quotes if it contains whitespace or special characters

i   Description

k   CKSUM hash value (see the @CKSUM function)

m   MD5 hash value (see the @MD5 function)

q   File or directory owner (NTFS only)

r   CRC32 hash value (see the @CRC32 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.) Note that you cannot get the allocated size on FTP servers or network sharenames.
       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). Note that the size will be truncated, not rounded.

@function[*]

call the specified variable [function](#) (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. Some fields cannot be reduced below a minimum width (for example, the **z** (size) field is a minimum of 15 digits).

If a PDIR line is empty (for example, if you have an embedded @IF), it will not be displayed.

If you want to append fields with no intervening whitespace, or with a custom delimiter character, you can use double quotes to specify arguments. For example, to display the date and time with no space between them:

```
pdir /(dymd""thms) *
```

Or to display the date and time separated by a +:

```
pdir /(dymd"+"thms) *
```

PDIR sets three internal variables:

%_pdir_dirs	The number of directories created
%_pdir_files	The number of files moved
%_pdir_errors	The number of errors

Example

To display the CRC, the full LFN and the owner of each file:

```
pdir /(r fpn q) *
```

Options:

Options on the command line apply only to the filenames which follow the option, and options at the end of the line apply to the preceding filename only. This allows you to specify different options for different groups of files, yet retains compatibility with the traditional [DIR](#) command when a single filename is specified.

Most options are used to select the desired files/directories. (This is in contrast to the DIR command.) The special option /(...) is used to specify which characteristics of the selected files or directories should be displayed in which sequence and format.

/	Display directory names with a trailing \\.
/A: . . .	Display only those files that have the specified attribute(s) set. See Attribute Switches for information on the attributes which can follow **/A: .** You can specify **/A:=** to display a dialog to help you set individual attributes.
/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**, PDIR will show the full path of each file instead of simply its name and extension. If you use **/B** with **/X** on an LFN drive, PDIR will display the short name of each file instead of the long name. **/B** also sets **/H.** /B1 will display relative paths when used with /S. (Normally, /B shows the full pathname for the file.)
/CD:	Define a custom directory colorization string to use instead of the COLORDIR environment variable, or the ColorDir option in TCMD.INI.
/D	Don't colorize the directory listing. See DIR for more information on directory colorization.
/H	Show the "**.**" and "**..**" directory names (normally suppressed).
/HL	Show hard links.
/I"*text*"	Select filenames by matching text in their descriptions. See Description Ranges 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
        f   Suppress bytes free in the footer
        h   Suppress the header
        j   Skip junctions (when used with /S)
        l   Don't display link name for symbolic links
        m:n Display a maximum of *n* directory entries
        s   Suppress the footer
        v   Suppress the volume label in the header
        z   Skip system directories (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
t	Same as **d**
u	Unsorted
x	When combined with /S, sorts the results from all directories together and displays them in a single listing. Note that /O:x will turn off headers and footers.

/P[*n*]   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. The /P option has an optional argument *n* that specifies the number of seconds to wait for a keystroke before continuing.

/Q   Show the owner of the file.

/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.

If you specify a **+** followed by a number after the /S, PDIR will not display any filenames until it gets to that depth in the subdirectory tree. For example, if you have a directory tree \a\b\c\d\e, /S+2 will not display the contents of \a or \a\b.

/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 command. If you specify multiple **/(...)** options, PDIR will insert a space in the output between each one.

4.2.128 PEE

Purpose: Copy standard output to multiple secondary commands via pipes

Format: **PEE** [/A /D /F"*format*" /R /T] "*app*" ...

        app   One or more applications that will receive the standard output.

/A(ppend)   /R (STDERR)
/D(ate)   /T(ime)
/F"..." (format)

See also: Y, piping and redirection.

Usage:

PEE is similar to TEE, but instead of redirecting STDOUT to multiple files, it redirects it to multiple secondary commands via pipes. You must enclose each command (and any arguments) in double quotes.

If you are typing at the keyboard to produce the input for P**EE**, you must enter a **Ctrl-Z** to terminate the input.

See Piping for more information on pipes.

Options:

/A   Append to the file(s) rather than overwriting them.

/D   Prefix each line with the current date (in yyyy-mm-dd format).

/F"..."   The *format* string. See @DATEFMT for details on *format* arguments.

/R   Redirect to STDERR instead of STDOUT.

/T   Prefix each line with the current time (in hh:mm:ss.ms format).

4.2.129 PIPEVIEW

Purpose: View realtime activity in a pipe

Format: PIPEVIEW [/D /E /GB /R /T /VH /X]

/D(ate)   /T(ime)
/E(nd)   /VH (text+hex)
/GB (greenbar)   /X (hex)
/R (STDERR)

Usage:

PIPEVIEW will read from STDIN, and display it in a VIEW window while also forwarding it on to STDOUT to be read by the next app. For example:

```
dir /s | pipeview | sort
```

Options:

/D	Prefix each line with the current date
/E	Always show the end of the pipe (most recent activity). Otherwise PIPEVIEW will default to showing the beginning of the pipe buffer.
/GB	(GreenBar) Display alternate shaded lines to make reading the output easier with long lines
/R	Write to STDERR instead of STDOUT
/T	Prefix each line with the current time
/VH	The pipe contents are displayed with each line of text followed by two lines containing the hex codes of each character.
/X	Display the pipe contents in hex

4.2.130 PLAYAVI

Purpose:	Play Windows .AVI (video clip) files
Format:	PLAYAVI [/A /C /S /*Vn*] *file*...

file	The file(s) to play

/A(synchronous)	/S(ynchronous)
/C(enter)	/V(olume)

File Selection

Supports extended wildcards, multiple file names, @file lists, and include lists.

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-RT 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 TCC-RT 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-RT pauses until the file has finished playing and its window closes.

/V Sets the volume level. The range is 0 (silent) to 100.

4.2.131 PLAYSOUND

Purpose: Play MP3, .WAV, Midi, and other sound files

Format: PLAYSOUND [/A /M /S /U /V*n*] *filename*

　　　　　filename The file to play

　　　　　/A(synchronous) /U(n mute)
　　　　　/M(ute) /V(volume)
　　　　　/S(ynchronous)

File Selection

Supports extended wildcards, multiple file names, @file lists, and include lists.

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-RT** waits for the sound file to complete and its window to close before continuing with the next command in a batch file or alias, or prompting you for a new command. You can change this default behavior with the /A switch, described below.

You can cancel the playing of a synchronous sound file by pressing Ctrl-Break while it is playing.

Options:

/A Plays the sound file in asynchronous mode. Control returns to TCC-RT 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-RT 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.

4.2.132 PLUGIN

Purpose: Load, unload, or display information about plugins

Format: PLUGIN [/B /C /F /I /K /L /P[*n*] /U /V] *plugin* ...

[/B (full pathname)](#)   [/L(oad)](#)
[/C(ommands)](#)   [/P(ause)](#)
[/F(unctions)](#)   [/U(nload)](#)
[/I(nfo)](#)   [/V(ariables)](#)
[/K(eystrokes)](#)

Usage:

Plugins allow you to write your own internal variables, variable functions, and internal commands, put them in a DLL, and have TCC-RT load them at startup. Plugin names will override existing internal names, so you can extend and/or replace internal variables and commands. When TCC-RT starts, it will automatically load any plugins in the default directory (the subdirectory PLUGINS\ in the TCC-RT installation directory). The plugins will be loaded before the startup file ([TCSTART](#)) 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.

You can specify a particular plugin to execute by prefixing the function / variable / command name with the plugin name and a $. For example:

```
echo %_myplugin$variable
echo %@myplugin$func[abc]
myplugin$mycommand
```

See the Plugin SDK for more information on developing plugins.

Options:

/B	Display the full pathnames of the plugins.
/C	Only display internal commands in the plugins.
/F	Only display variable functions in 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.
/K	Only display keystroke plugins.
/L	Loads the specified plugins. If the filename is *****, load all plugins from the default directory (the subdirectory PLUGINS\ in the TCC-RT installation directory).
/P[*n*]	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](#). The /P option has an optional argument *n* that specifies the number of seconds to wait for a keystroke before continuing.
/U	Unloads the specified plugin. If the filename is *****, unloads all plugins.

/V   Only display internal variables in the plugins.

4.2.133 POPD

Purpose: Restore the disk drive and directory at the top of the directory stack

Format: POPD [/X * *n*] [*name*]

> **_n_**   The number of directories to pop
> **_name_**   Directory name to match

/X (exclude)

See also: DIRS, PUSHD, and @DIRSTACK.

Usage:

Each time you use the PUSHD 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.

This example saves and changes the current disk drive and directory with PUSHD, and then restores it. The current directory is shown in the prompt:

```
[c:\] pushd d:\database\test
[d:\database\test] pushd c:\wordp\memos
[c:\wordp\memos] pushd a:\123
[a:\123] popd
[c:\wordp\memos] popd
[d:\database\test] popd
[c:\]
```

You can use the DIRS command to see the complete list of saved drives and directories (the directory stack).

The POPD command followed by an asterisk [*] clears the directory stack without changing the current drive and directory.

If the directory on the top of the stack is not on the current drive, POPD will switch to the drive and directory on the top of the stack without changing the default directory on the current drive.

You can optionally restore only the most recent directory in the stack which matches a name. For example:

> POPD c: Pop the most recent directory on C:
> POPD \\server\share   Pop the most recent directory on the UNC share

The name to match can include wildcards.

Note that this means you can optionally choose to POPD to any directory in the directory stack, not just the most recent one.

Options:

/X   Don't save the current directory to the Directory History list.

4.2.134 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.

If ***title*** begins with a **=**, it is assumed to be a process ID instead of a title. (Note that this is less reliable than providing a title, as a process can have multiple top-level windows.)

See the Windows SDK documentation for a list of possible messages and their parameters.

4.2.135 POWERMONITOR

Purpose: Monitor system power changes

Format: POWERMONITOR [/C [*action*]]
            POWERMONITOR [Battery | AC | DC | Scheme | Display | Resume | Suspend] [*n* | FOREVER] *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)

Usage:

POWERMONITOR monitors power scheme change, battery power, AC / DC switch, system suspend, and system resume. Note that Windows will send an immediate notification for the current scheme, AC/DC, and battery.

The command line will be parsed and expanded before POWERMONITOR 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 the last argument on the line is a single **(,** it is interpreted as the beginning of a command group. POWERMONITOR will append the following lines (in a batch file) or prompt you for more input (at the command line) until it gets a closing **).**

If you don't enter any arguments, POWERMONITOR 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-RT thread is also performing I/O. You may need to use START or DETACH in **command** to avoid conflicts.

POWERMONITOR creates four environment variables on a power change that can be queried by **command**. The variables are deleted after **command** is executed.

       _powerbattery - returns the battery % (0-100).
       _powersource - returns the power source (AC or DC).
       _powerdisplay - returns 0 if the primary monitor is powered off or 1 if it is on.
       _powerscheme - returns the power scheme in use:
            0 - Power Saver
            1 - Maximum Performance
            2 - Balanced
            3 - Unknown

Example:

If you want to be alerted whenever the system switches to DC power:

```
powermonitor DC forever echo just switched to battery power!
```

Options:

    /C   If **name** is specified, remove the monitor for that power action. Otherwise, remove all active power monitors.

4.2.136 PRINT

Purpose: Print the specified file(s) using the application associated with each file's extension

Format: PRINT [/A *printer* /D *printer* /S *printer*] *filename* ...

       /A(dd) printer   /S(et) default printer
       /D(elete) printer

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 accepts piped & redirected input to send to the printer. If there is no *filename*, PRINT will read from STDIN, create a temporary file, and send it to the printer.

PRINT depends on proper Windows File Associations 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.

/S   Set the default printer.

4.2.137 PRINTF

Purpose: Display a formatted string using the C Printf format.

Format: PRINTF "*format string*" args ...

Usage:

The arguments following the format string will be inserted in the output string according to the format type in the format string. The arguments can be variable names, variable functions, or literal strings; i.e.:

PRINTF "%s %d %x" %var1 999 %hexvar

The *format type* syntax is:

%[flags][width][.precision][length]type

flags	description
_	Left-justify within the given field width; Right justification is the default (see *width* sub-specifier).
+	Prefix the result with a plus or minus sign (+ or -) even for positive numbers. By default, only negative numbers are preceded with a - sign.
0	Prefix the number with zeroes (0) instead of spaces when padding is specified (see *width* sub-specifier).

width	description
number	Minimum number of characters to be printed. If the value to be printed is less than this number, the result is padded with spaces.
*	The *width* is not specified in the *format* string, but as an additional integer argument preceding the argument to be formatted.

.precision	description
`.number`	For integer specifiers (d, i, o, u, x, X): *precision* is the minimum number of digits to be written. If the value to be written is less than *precision*, the result is padded with leading zeros. For f and g specifiers: The maximum number of significant digits to be printed.
`.*`	The *precision* is not specified in the *format* string, but as an additional integer value argument preceding the argument that has to be formatted.

Type	Output
`d or i`	Signed decimal integer
`u`	Unsigned decimal integer
`x`	Unsigned hexadecimal integer
`X`	Unsigned uppercase hexadecimal integer
`f or g`	Decimal floating point
`c`	Character
`s`	String
`%`	A % followed by another % will write a single %

If you prefix a type with an **L**, PRINTF will insert commas as thousands separators. For example:

    PRINTF "%Ld" 123456789

will output:

    123,456,789

4.2.138 PRIORITY

Purpose: Display or set process priority, or suspend or resume a process

Format: PRIORITY [/D /E /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

> /D(isable)   /R(esume)
> /E(nable)   /S(uspend)
> /Q(uiet)

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-RT** process.

If you only provide a PID or window title, PRIORITY will display the current priority.

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).

Example:

Set the process with the window title beginning with TC28 to high priority:

```
priority "TC28*" HIGH
```

Options:

/D	Disable the ability of Windows to temporarily boost the priority of threads in the process.
/E	Enable the ability of Windows to temporarily boost the process of threads in the process.
/Q	Don't display any suspend / resume messages.
/R	Resume the process.
/S	Suspend the process.

4.2.139 PROCESSMONITOR

Purpose:	Monitor process start or end
Format:	PROCESSMONITOR [/C [*name*]] PROCESSMONITOR *name* STARTED | ENDED | HUNG *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)](#)

Usage:

The process name can include wildcards. If you do not include a path for ***name***, PROCESSMONITOR will only compare the filename part of the process names.

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 the last argument on the line is a single **(**, it is interpreted as the beginning of a command group. PROCESSMONITOR will append the following lines (in a batch file) or prompt you for more input (at the command line) until it gets a closing **).**

HUNG will test the process's main window to see if it is still responding to messages. If the process doesn't respond or call GetMessage within 5 seconds, the condition will be triggered. (This is normally only useful for GUI apps.)

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-RT thread is also performing I/O. You may need to use START or DETACH 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.

4.2.140 PSHELL

Purpose: Execute a PowerShell script or string

Format: PSHELL [/C /S *script* ...]

/C(lose)
/S(tring)

See also @PSHELL.

Usage:

Note that you may need to enable PowerShell scripting on your system; on recent versions of Windows it is disabled by default (for security).

PSHELL supports multiple string or filename arguments. If a string or filename has embedded whitespace, you must enclose it with double quotes. For example:

```
pshell /s "type $Profile"
```

Without the double quotes PSHELL would interpret this as two commands.

Option:

/C	Close the persistent PowerShell interpreter
/S	Execute a string (like @PSHELL)

4.2.141 PSUBST

Purpose: Associate a path with a drive letter

Format: PSUBST [*drive1*: [*path*]]
PSUBST /D *drive1*:
PSUBST /P *drive1*: [*path*]

drive1:	Specifies a virtual drive to which you want to assign a path.
path	Specifies a path you want to assign to a virtual drive (no trailing backslash).

/D(elete)
/P(ersist)

Usage:

PSUBST works like the Windows SUBST command, but the drive substitution is persistent (i.e., when the machine is restarted).

PSUBST with no parameters will display a list of the current virtual drives. If a drive is persistent, it will be prefixed with a *.

Because PSUBST needs to write to the HKLM registry hive, PSUBST must be run in an elevated TCC-RT session.

Options:

/D	Deletes a substituted (virtual) drive.
/P	Make a new or existing virtual drive persistent.

4.2.142 PUSHD

Purpose: Save the current disk drive and directory, optionally changing to a new drive and directory

Format: PUSHD [/R /X *path*]

path	The name of the new default drive and directory.

/R(eparse point)
/X (exclude)

See also: DIRS, POPD, and @DIRSTACK.

Usage:

PUSHD saves the current drive and directory to a "last in, first out" directory stack. The POPD command returns to the last drive and directory that was saved by PUSHD. You can use these commands together to change directories, perform some work, and return to the starting drive and directory. The DIRS command displays the contents of the directory stack.

To save the current drive and directory, without changing directories, use the PUSHD command by itself, with no *path*.

If a *path* is specified as part of the PUSHD command, the current drive and directory are saved and PUSHD changes to the specified drive and directory. If the *path* includes a drive letter, PUSHD changes to the specified directory on the new drive without changing the current directory on the original drive.

PUSHD supports Windows shell folder names; see CDD for details.

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 for details).

The directory stack can hold up to 16383 characters, or about 500+ typical entries (depending on the length of the names). If you exceed this limit, the oldest entry is removed before adding a new entry.

Example:

Save the current directory and change to C:\WORDP\MEMOS, then return to the original directory:

```
[c:\] pushd \wordp\memos
[c:\wordp\memos] popd
[c:\]
```

Options:

/R   Change to the target of the reparse point (hard link or symbolic link).

/X   Don't save the current directory to the Directory History list.

4.2.143 QUERYBOX

Purpose: Pops up a dialog box to get an input string from the user and save it in an environment variable

Format: QUERYBOX [/CUE="*text*" /D /E /L*n* /P /POS=*top,left* /T*n*] ["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.
/CUE	Cue text to display in the input box

/D(igits only)   /P(assword)
/E(dit existing value)   /POS (ition)
/L (maximum Length)   /T(imeout)

See also: INKEY, INPUT, and MSGBOX.

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.

The /CUE option displays the cue text in light gray in the input box (it disappears as soon as you enter a character).

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.

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.

QUERYBOX returns a value of zero in the internal variable %_? after a successful operation, and a non-zero value otherwise (a timeout returns 20, a cancel returns 2). Be sure to save the return value in another variable or test it immediately; because the value of **%_?** changes with every internal command.

Example:

To prompt for a string and store it in the variable NAME:

```
querybox "File Name" Enter a name:  %%name
```

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-RT window.

/Tn   Wait for a maximum of *n* seconds for a response.

4.2.144 QUIT

Purpose: Terminate the current batch file

Format: QUIT [*value*]

value   The numeric exit code to return to TCC-RT or to the previous batch file.

See also: CANCEL and EXIT.

Usage:

QUIT provides a simple way to exit a batch file before reaching the end of the file. If you QUIT a batch file called from another batch file, you will be returned to the previous file at the line following the original CALL.

QUIT only ends the current batch file. To end all batch file processing, use the CANCEL command.

If you specify a *value*, QUIT will set the ERRORLEVEL or exit code to that value. For information on exit codes see the IF command, and the %? variable. Numeric input may be entered in either decimal format (a sequence of 0-9 digits) 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.

Example:

This batch file fragment checks to see if the user entered "quit" and exits the batch file if true.

```
input  Enter your choice :  %%option
if  "%option" == "quit"  quit
```

4.2.145 RANDOM

Purpose: Generate a random integer, bool, float, char, or string.

Format: RANDOM [/I start, end] /[B] [/F start, end, precision] [/C start, end] [/S start, end, length] [/D n]

Usage:

RANDOM can generate a variety of random output - integers, booleans, floating point numbers, characters, or strings.

See also @RANDOM.

Examples:

To create a random integer between 5 and 15:

random /i 5, 15

To create a random float between 1.7 and 2.6, to 5 decimal places:

random /f 1.7, 2.6, 5

Options:

   /B - Create a boolean (0 or 1)

/C *start, end* - Create a character between (inclusive) the characters *start* and *end*.

/D *n* - Roll an *n* sided dice.

/F *start, end, precision* - Create a floating point number >= *start* and <= *end,* with *precision* decimal places. *Start* and *end* can be integers or floats.

/I *start, end* - Create an integer >= *start* and <= *end*. The *start* and *end* arguments are signed 64-bit values.

/S *start, end, length* Create a string composed of characters between *start* and *end* (inclusive).

4.2.146 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 for details).

/I (match descriptions)	/Q(uiet)
/K (no Recycle Bin)	/R(ecycle bin)
/N (disable options)	/S(ubdirectories)

See also: MD.

File Selection

Supports 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 Servers.

Usage:

RD and RMDIR are synonyms. You can use either one.

RD removes directories from the directory tree. For example, to remove the subdirectory MEMOS from the subdirectory *WP*:

```
rd \wp\memos
```

Before using RD, you must delete all files and subdirectories (and their files) in the **path** you want to remove. Remember to remove hidden and read-only files as well as normal files (you can use DEL /Z to delete hidden and read-only files).

You can use wildcards in the **path**.

When removing a directory on an LFN drive, you must quote any *path* which contains white space or special characters.

If RD deletes one or more directories, they will be deleted from the extended directory search 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.

You can remove directories on FTP servers. For example:

```
rd ftp://ftp.abc.com/data
```

RD sets two internal variables:

%_rd_dirs	The number of directories deleted
%_rd_errors	The number of errors

(Note that if you do an RD /S, the actual deletions are done by DEL, so check the DEL variables.)

Options:

/I"text"   Select directories by matching text in their descriptions. The text can include wildcards and extended wildcards. The search text must be enclosed in 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 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 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 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, 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.

4.2.147 REBOOT

Purpose: Reboot the computer, log off Windows, or shut down

Format: REBOOT [/= /A /B *text*] /F /H /K /L /M[0|1] /P /R /S /V /W /Y]

/A (restart apps)   /P(ower off)
/B(lock)   /R(eboot)
/F(orce)   /S(hutdown)
/H(ibernate   /V(erify)
/K (lock)   /W (standby)
/L(ogoff)   /Y (hybrid)
/M(onitor)

Usage:

REBOOT will log off or shut down the operating system, or completely restart your computer. It normally performs a warm reboot, or 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-RT 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:

/=   Display the REBOOT command dialog to help you set the command line options. The /= option can be anywhere on the line; additional options will set the appropriate fields in the command dialog.

A   Restart applications. Shuts down the system and then restarts it, as well as any applications that have been registered for restart using the Windows RegisterApplicationRestart API.

/B text Block shutdown / reboot. The system will display *text* in the popup explaining the reason for blocking the shutdown.

/F   Force the specified reboot option. This option does not send the WM_QUERYENDSESSION message to applications, so this can cause applications to lose data. This option is only valid when used with the /P, /R, or /S 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.
/Y	**H**ybrid shutdown - prepare the system for a faster reboot. This option is only valid wihen used with the /P, /R, or /S options.

4.2.148 RECYCLE

Purpose:	Delete files in the recycle bin or display the recycle bin status
Format:	RECYCLE [/D /E /Q /P] [*drives* ...]

 drives   Local fixed and removable (non CD-ROM / DVD) drives

/D(elete)   /P(rompt)
/E (no error messages)   /Q(uiet)

Usage:

If you don't specify any drives, RECYCLE will display the recycle bin status, or if /D is specified delete everything in the recycle bin for all local drives.

RECYCLE will empty the recycle bin for an entire drive; there is no way to specify individual files.

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 (at least one drive must be specified).
/Q	Don't display the name of the recycle bin(s). This option is most often used in batch files.

4.2.149 REGDIR

Purpose:	Display the specified Windows Registry tree
Format:	REGDIR [/D /F /Nb /P[*n*] /S*n* /T /TS /V /X] *keyname*

 keyname   The Windows Registry key to enumerate

/D(ata)   /T(ime stamp)
/F(ull name)   /TS (include seconds)
/Nb (no REG_BINARY)   /V(alues)

[/P(ause)](#)   [/X (hex)](#)
[/Sn (nesting depth)](#)

Usage:

REGDIR will display the Windows Registry like TREE or DIR does with the file system. The Windows Registry is **very** large, so trying to display something like:

```
regdir /v /d hkcu\software
```

will typically display tens of thousands of records, and can potentially run out of memory in 32-bit Windows.

The key must begin with either the full root key or the short name:

Full root key	Short
HKEY_CLASSES_ROOT	HKCR
HKEY_CURRENT_USER	HKCU
HKEY_LOCAL_MACHINE	HKLM
HKEY_USERS	HKU
HKEY_CURRENT_CONFIG	HKCC
HKEY_PERFORMANCE_DATA	HKPD

Option:

/D Display the data for all values (only valid when used with /V)

/F Display the full name for each key. (The default is to display only the indented name of the current key, similar to TREE's output.)

/Nb Do not display the contents of REG_BINARY values.

/P[*n*] Pause after displaying each page. The /P option has an optional argument *n* that specifies the number of seconds to wait for a keystroke before continuing.

/S REGDIR will limit the nesting recursion to that number. REGDIR defaults to unlimited key recursion.

/T Prefix the key names with the time stamp of their last change in hh:mm format.

/TS Prefix the key names with the time stamp of their last change in hh:mm:ss format.

/V Display the values for each key.

/X Display the REG_DWORD, REG_DWORD_BIG_ENDIAN, and REG_QWORD values in hex. Only valid when used with /V and /D.

4.2.150 REGMONITOR

Purpose: Monitor Windows Registry keys

Format: REGMONITOR [/C [*key*]]
REGMONITOR *key* NAME ATTRIBUTES VALUE SECURITY *n command*

    key   Key name

NAME	Subkey added or deleted
ATTRIBUTES	Changes to the key attributes (such as the security descriptor information)
VALUE	Changes to the value of a key
SECURITY	Changes to the security descriptor
n	Number of repetitions (or **FOREVER**)
command	Command to execute when condition is triggered

/C(lear)

Usage:

The command line will be parsed and expanded before REGMONITOR 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 the last argument on the line is a single **(,** it is interpreted as the beginning of a command group. REGMONITOR will append the following lines (in a batch file) or prompt you for more input (at the command line) until it gets a closing **).**

If you don't enter any arguments, REGMONITOR will display the registry keys it is currently monitoring.

The key must begin with either the full root key or the short name:

Full root key	**Short**
HKEY_CLASSES_ROOT	HKCR
HKEY_CURRENT_USER	HKCU
HKEY_LOCAL_MACHINE	HKLM
HKEY_USERS	HKU
HKEY_CURRENT_CONFIG	HKCC

If you append a ***** to the key, REGMONITOR will monitor the specified key and all of its subkeys.

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-RT thread is also performing I/O. You may need to use START or DETACH in *command* to avoid conflicts.

Example:

```
regmonitor "HKCU\Software\JP Software\Take Command 28\*" name value
forever echo Windows Registry updated!
```

Options:

/C If *name* is specified, remove the monitor for that registry key. Otherwise, remove all registry monitors.

4.2.151 REM

Purpose: Put a comment in a batch file

Format: REM [*comment*]

comment The text to include in the batch file.

See also: COMMENT.

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. TCC-RT 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 CMD. A simpler method for creating a zero-byte file with TCC-RT is to use **>filename** as a command, with no actual command before the [**>**] redirection character.)

4.2.152 REN / RENAME

Purpose: Rename files or subdirectories

Format: REN [/A:[[-][+]rhsadecijopt] /B /E /I"text" /N[enst] /O:[-]adegnrstu /P /Q /S /T] [@*file*]
          old_name... new_name
             or
          RENAME [/A:[[-][+]rhsadecijopt] /E /I"text" /N[enst] /O:[-]adegnrstu /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 for details).

/A: (Attribute select)	/O:... (Order)
/B (Rename on reboot)	/P(rompt)
/E (No error messages)	/Q(uiet)
/I"text" (match description)	/S(ubdirectory)
/MD (Create target directory)	/T(otal)

/N (Disable)

See also: COPY and MOVE.

File Selection:

Supports attribute switches, extended wildcards, ranges, multiple file names, delayed variable expansion, and include lists. Use wildcards with caution on LFN volumes; see LFN File Searches for details.

Internet: Can be used with FTP/HTTP Servers 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. New files may be on different file systems or drives; new directories must be on the same drive.

In its simplest form, you give REN the ***old_name*** of an existing file or subdirectory and then a ***new_name***. The ***new_name*** must not already exist; you can't give two files the same name (unless they are in different directories). The first example renames the file *MEMO.TXT* to *MEM.TXT*. The second example changes the name of the *\WORD* directory to *\WP*:

```
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**.

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 and IFTP.

REN supports regular expression back references in the target name. If you are using back references, you must also use a regular expression in the source name. The syntax is:

```
ren ::filename ::target
```

REN sets three internal variables:

%_ren_dirs	The number of directories renamed
%_ren_files	The number of files renamed
%_ren_errors	The number of errors

Note: The wildcard expansion process will attempt to allow both CMD-style "extension" matching (assumes only one extension, at the end of the word) and the advanced TCC-RT 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 for information on the attributes which can follow **/A:**. Do not use /A: with @file lists. See @file lists for details.

You can specify **/A:=** to display a dialog to help you set individual attributes.

/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 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 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 or more of the following arguments has an alternate meaning:

e	Don't display errors.
n	Don't update the file descriptions
s	Don't display the summary

/O:... Sort the files before processing.

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, instead of numerically when numeric substrings are included in the name or extension.
d	Sort by date and time (oldest first); also see **/T:acw**
e	Sort by extension
g	Group subdirectories first, then files
r	Reverse the sort order for all options
s	Sort by size
t	Same as **d**
u	Unsorted

The /O:... option saves all of the matching filenames and then performs the rename. This avoids the potential problem of renaming files more than once.

/P Prompt the user to confirm each rename operation. Your options at the prompt are explained in detail under Page and File Prompts.

/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 REN.

/T   Don't display filenames as they are renamed, but report the number of files renamed. See also **/Q**.

4.2.153 REPEAT

Purpose: A simpler way than DO or FOR to execute a counted loop.

Format: REPEAT *n command ...*

         n - The number of times you want to repeat *command*.

Usage:

To run the command *test* 10 times:

```
repeat 10 testcommand
```

REPEAT sets the internal command variable **_repeat** to the current loop counter (1 to *n*).

4.2.154 RESOLUTION

Purpose: Change the resolution of the specified display

Format: RESOLUTION [*displayname*] *width height* [*depth* [*frequency*]]

displayname	The name of the monitor to update
width	The new display width in pixels
height	The new height in pixels
depth	The new color depth
frequency	The new refresh frequency

Usage:

If you don't specify any arguments, RESOLUTION will display the display devices and monitors.

4.2.155 RESTOREPOINT

Purpose: Create, remove, or list Windows system restore points.

Format: RESTOREPOINT [/C /D=*description* /R *n*]

         /C(reate)
         /D(escription)
         /R(emove)

Usage:

TCC-RT must be running in an elevated session to create, remove or list restore points. RESTOREPOINT is not supported in Server versions of Windows.

If you don't specify any arguments, RESTOREPOINT will display the existing restore points. If there are no restore points, RESTOREPOINT will not display anything (including an error).

If you have disabled restore points, RESTOREPOINT will return a (Windows) error.

Example:

Create a restore point named "July-2021":

```
restorepoint /c /d=July-2021
```

Options:

/C   Create a restore point

/D=   The description shown when displaying restore points

/R   Remove the restore point whose sequence is *n*.

4.2.156 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.

Usage:

TCC-RT 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-RT will display a "Missing GOSUB" error message.

You cannot execute a RETURN from inside a DO loop.

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 _? variable.

Example:

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
      .
```

4.2.157 REXEC

Purpose: Remotely execute commands

Format: REXEC [/H *host* /U *name* /P *password* /IPv6 /Rn /Tn] *host* [/L *userid*] *command* ...

command   The command to execute

/H(ost name)   /R(emote port)
/IPv6   /T (firewall type)
/L (user ID)   /U(sername)
/P(assword)

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, REXEC 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 rexec service, so you will need to get one from a third-party and install it on the remote system before executing REXEC.

Options:

/H   Firewall host name

/IPv6   Use IPv6 instead of IPv4

/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

4.2.158 RSHELL

Purpose: Remotely execute commands

Format: RSHELL [/H *host* /U *name* /P *password* /IPv6 /R*n* /T*n*] host [/L *userid*] *command* ...

>   ***command***   The command to execute

/H(ost name)   /R(emote port)
/IPv6   /T (firewall type)
/L (user ID)   /U(sername)
/P(assword)

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

/IPv6   Use IPv6 instead of IPv4

/L   User name

/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

4.2.159 SAVECONSOLE

Purpose:	Save the console screen buffer to a file

Format:	SAVECONSOLE [/H /T /W] *filename* ...

/H(tml)
/T (header)
/W(hitespace)

Usage:

SAVECONSOLE can save the screen buffer either in plain text format or as an HTML file, including colors.

Example:

Save the console buffer as an HTML file named "tcc_console.html", with a header:

```
saveconsole /H /T tcc_console.html
```

Option:

/H	Save the console as an HTML file

/T	Include a header with the console title + date + time

/W	Don't strip trailing white space on each line (this will result in a much bigger file)

4.2.160 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, ECHOS and ECHOSERR, SCRPUT, TEXT, and VSCRPUT.

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 or VSCRPUT. SCREEN always leaves the cursor at the end of the displayed text.

The *row* and *column* values are zero-based, so on a 25 line by 80 column display, valid *rows* are 0 - 24 and valid *columns* are 0 - 79. SCREEN checks for a valid *row* and *column*, and displays a "Usage" error message if either value is out of range.

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-RT window:

```
screen 999 999 Hello, World
```

4.2.161 SCREENMONITOR

Purpose: Monitor the Windows screen saver

Format: SCREENMONITOR [/C]
                SCREENMONITOR *n command*

>   *n*   Number of repetitions (or **FOREVER**)
>   *command*   Command to execute when the Windows screen saver is activated

>   /C(lear)

Usage:

SCREENMONITOR will set its trigger when the Windows screen saver is activated.

If you don't enter any arguments, if SCREENMONITOR is active it will display the repeat count and the command.

The command line will be parsed and expanded before SCREENMONITOR 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 the last argument on the line is a single **(**, it is interpreted as the beginning of a command group. SCREENMONITOR will append the following lines (in a batch file) or prompt you for more input (at the command line) until it gets a closing **)**.

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-RT thread is also performing I/O. You may need to use START or DETACH in **command** to avoid conflicts.

Options:

/C   Remove the screen saver monitor.

4.2.162 SCRIPT

Purpose: Run a script using an Active Scripting engine

Format: SCRIPT [/E *engine*] [*filename* ...]

/E(ngine)

engine   The name of the scripting engine

Usage:

If you don't specify any arguments, SCRIPT will display the installed engines.

You can call internal TCC-RT commands from any Active Scripting language using the tcommand() interface created by SCRIPT. For example, create a JavaScript file named *testjs.js*:

```
var d1 = "First Message";
var d2 = "echo Second Message";
var d3 = "dir /w";
TakeCommand.msgbox(d1);
TakeCommand.tcommand(d2);
TakeCommand.tcommand(d3);
```

You can then pass *testjs.js* to SCRIPT:

```
script testjs.js
```

See also the @SCRIPT variable function.

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.

4.2.163 SCRPUT

Purpose: Position text on the screen and display it in color

Format: SCRPUT *row col* [/C /U] [BRIght] *fg* ON [BRIght] *bg text*

row   Starting row
col   Starting column
fg   Foreground character color
bg   Background character color

> ***text*** The text to display
>
> /C (move cursor)
> /U (move to end of string)

See also: ECHO and ECHOERR, ECHOS and ECHOSERR, SCREEN, TEXT, and VSCRPUT.

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 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-RT window. The maximum ***column*** value is determined by the current virtual screen width.

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-RT 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***.

Example:

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
```

Options:

> **/C** Move the cursor to the specified position after writing the string.
>
> **/U** Move the cursor to the end of the string.

4.2.164 SELECT

> ***Purpose***: Interactively select files for a command

Format: SELECT [/1 /A[[:][-][+]rhsadecijopt] /C /D /E /H /I"*text*" /J /L /O:[-]
acdeginorstuz /Q /T:acw /X /Z] [*command*] ... (*files*...)...

         command   The command to execute with the selected files.
         files   The files from which to select. File names may be enclosed in either
                                      parentheses or square brackets. The difference is explained below.

/1 One selection only	/J(ustify names)
/A(ttribute select)	/L(ower case)
/C(ompression)	/O(rder)
/D(isable color coding)	/Q (owner)
/E (use upper case)	/T(ime)
/H(ide dots)	/X (display short names)
/I"text" (match descriptions)	/Z (FAT format)

File Selection

Supports extended wildcards, ranges, multiple file names, and include lists. Ranges **must** appear immediately after the SELECT keyword.

Internet: Can be used with FTP servers. See Using FTP/HTTP Servers.

Usage:

SELECT allows you to select files for internal and external commands by using a "point and shoot" display. You can have SELECT execute a command once for each file you select, or have it create a list of files for a command to work with. The ***command*** can be an internal command, an alias, an external command, or a batch file.

If you use parentheses around the ***files***, SELECT executes the ***command*** once for each file you have selected. During each execution, one of the selected files is passed to the ***command*** as 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.

If the last argument on the line is a single **(,** it is interpreted as the beginning of a command group. SELECT will append the following lines (in a batch file) or prompt you for more input (at the command line) until it gets a closing **).**

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 and IFTP.

SELECT will colorize the directory listing. See DIR for more information on directory colorization.

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. 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](). 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 (*.cmd *.exe) q:\
```

will let you select from among the *.CMD* files in the current directory, and will then invoke the COPY command to copy each file you select to the root of drive Q:. After the *.CMD* files are done, the operations will be repeated for the *.EXE* files.

If you want to select from a list of all the *.CMD* and *.EXE* files mixed together, create an [include list]() inside the parentheses by inserting a semicolon:

```
select copy (*.cmd;*.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 [*.cmd;*.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. 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. 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\*.cmd;*.exe;*.btm;*.bat)
```

With the **/l** option, you can select files based on their descriptions. SELECT will display files if their description matches the text after the **/l** 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-RT 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:

/1   Only allow one selection.

/A[:]   Select only those files that have the specified attribute(s) set. See Attribute Switches for information on the attributes which can follow **/A:**.

        You can specify **/A:=** to display a dialog to help you set individual attributes.

/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 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 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.

/Q   Display the file owner (requires > 80 column display).

/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   (Obsolete) Display short filenames in FAT format (like **/Z**), on LFN drives. If you are passing the SFNs to a file-handling command (like COPY, DEL, REN, etc.) you will need to have the "Search for SFNs" directive set (see OPTION / Startup). That option is not set

by default, and without it TCC-RT will not find SFNs in normal directory searches on LFN / NTFS drives.

/Z (Obsolete) Display a directory on an LFN drive in the old-style format, with the filename at the left and the description at the right. Long names will be truncated to 12 characters; if the name is longer than 12 characters, it will be followed by a right arrow.

4.2.165 SENDHTML

Purpose:	Send an HTML-formatted email message

Format: SENDHTML [/A *file1* [/A *file2* ...] /D /E*address* /H"*header: value*" /I*n* /IPv6 /M /P*n* /R /SMTP=*server* /S*n* /SSL[=*n*] /USER=*address* /V /X] "*address*[,*address*...] [cc:*address*[,*address*] bcc:*address*[,*address*...]]" *subject* [*text* | @*msgfile*]

file1...	The attachment files
address	The destination email address
subject	The subject line
text	The message to send
msgfile	The file containing the message body
/SSL=n	SSL negotiation type
/SMTP=server	Override the default SMTP server
/USER=address	Override the default sending email account

/A file	Attachment	/M	CRAM-MD5 authentication
/D	Delivery Confirmation	/Pn	Priority
/E	Reply-to address	/R	Send read receipt
/H	Send custom header	/Sn	Sensitivity
/In	Importance	/V	Verbose
/IPv6	Use IPv6 instead of IPv4	/X	EHLO

See also: SNPP and SMPP.

Usage:

SENDHTML sends an HTML email message from TCC-RT via SMTP. The text of the message can be entered either on the command line or read from a text file. SENDHTML also supports SMTP over SSL.

Before you can use SENDHTML, you must either set the SMTP 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 SENDHTML.

A SENDHTML message has three required parts: an address, a subject, and message. Optionally it may also have attachments.

1. The ***address*** field contains one or more standard Internet email addresses:

        sendhtml 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:

```
sendhtml "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 SENDHTML 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:**):

```
sendhtml abc@xyz.com Party @c:\messages\invitation.txt
sendhtml 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:

```
sendhtml /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.

/I*n*   Set the Importance where ***n*** is:

1	High
2	Normal (default)
3	Low

/IPv6   Use IPv6 instead of IPv4.

/M   Use CRAM-MD5 authentication.

/P*n*   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

/SMTP Overrides the default SMTP server (as set in the registry) to use when sending mail.

/SSL=n Type of SSL negotiation. The values are:

    0 Automatic (default if no **n** value is specified). If the remote port is set to the standard plaintext port, SENDHTML will use Explicit mode. In all other cases, SSL negotiation will be implicit.
    1 Implicit - SSL negotiation will start immediately after the connection is established.
    2 Explicit - SENDMAIL will first connect in plaintext, and then explicitly start SSL negotiation.
    3 No SSL negotiation or security. (This is the default if /SSL is not specified.)

/USER Overrides the default email account (as set in the registry) to use when sending mail.

/V Show all the interaction with the server, except the message header and message body text.

/X Send EHLO instead of HELO.

4.2.166 SENDMAIL

Purpose: Send an email message

Format: SENDMAIL [/A *file1* [/A *file2* ...] /D /E*address* /H"*header*:
value" /I*n* /IPv6 /M /P*n* /R /S*n* /SMTP=*server* /SSL[=*n*] /USER=*address* /V /X]
"*address*[,*address*...] [cc:*address*[,*address*] bcc:*address*[,*address*...]]" *subject* [*text* |
@*msgfile*]

file1...	The attachment files
address	The destination email address
subject	The subject line
text	The message to send
msgfile	The file containing the message body
/SSL=n	SSL negotiation type
/SMTP=server	Override the default SMTP server
/USER=address	Override the default sending email account

/A file	Attachment		/M	CRAM-MD5 authentication
/D	Delivery Confirmation		/Pn	Priority
/E	Reply-to address		/R	Send read receipt
/H	Send custom header		/Sn	Sensitivity
/In	Importance		/V	Verbose
/IPv6	Use IPv6 instead of IPv4		/X	Send EHLO

See also: SNPP and SMPP.

Usage:

SENDMAIL sends an email message from TCC-RT via SMTP. The text of the message can be entered either on the command line or read from a text file. SENDMAIL also supports SMTP over SSL.

Before you can use SENDMAIL, you must either set the SMTP 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, a subject, and message. Optionally it may also have attachments.

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. For example, to send HTML mail:

      sendmail /h"Content-Type: text/html" ...

/In   Set the Importance where *n* is:

1	High
2	Normal (default)
3	Low

/IPv6   Use IPv6 instead of IPv4.

/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

/SMTP   Overrides the default SMTP server (as set in the registry) to use when sending mail.

/SSL=n   Type of SSL negotiation. The values are:

0	Automatic (default if no *n* value is specified). If the remote port is set to the standard plaintext port, SENDMAIL will use Explicit mode. In all other cases, SSL negotiation will be implicit.
1	Implicit - SSL negotiation will start immediately after the connection is established.
2	Explicit - SENDMAIL will first connect in plaintext, and then explicitly start SSL negotiation.
3	No SSL negotiation or security. (This is the default if /SSL is not specified.)

/USER   Overrides the default email account (as set in the registry) to use when sending mail.

/V   Show all the interaction with the server, except the message header and message body text.

/X   Send EHLO instead of HELO.

4.2.167 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)

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 the last argument on the line is a single **(,** it is interpreted as the beginning of a command group. SERVICEMONITOR will append the following lines (in a batch file) or prompt you for more input (at the command line) until it gets a closing **).**

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-RT thread is also performing I/O. You may need to use START or DETACH 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

Example:

Send an email if the service "mytestservice" stops:

```
SERVICEMONITOR mytestservice STOPPED sendmail bob@bob.com "Service
Stopped" The Windows service "mytestservice" stopped!
```

Options:

/C If *name* is specified, remove the monitor for that service. Otherwise, remove all service monitors.

4.2.168 SERVICES

Purpose: Display, stop, or start system services

Format: SERVICES [/I /P /R /S /T*n*] [*name* ...]

/I (PID)   /S(top service)
/P(ause)   /T*n* (type)
/R(un)

Usage:

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

You must be an admin user to run or stop a service.

Options:

/I Display the PID's for services. Note that stopped services will return 0 for the PID, as will Windows services.

/P Pause after displaying each page.

/R Run the specified service(s).

/S Stop the specified service(s).

/Tn The type of services to enumerate. This can be a combination (OR'd) of the following values:

 1 Kernel drivers
 2 File system drivers
 16 Services that run in their own process
 32 Services that share a process with one or more other services

4.2.169 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 /B /D /M /O /S /U /V /E /RO /R [*file*...] /T:*type* | *name=value* | *prompt*]

 Deletion mode:

SET [/D /M /S /U /V /E] *name*=

file	One or more input files from which to read variable definitions.
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	/R	Read from file(s)
/B	Batch variable	/RO	Readonly variable
/D	Default	/S	System
/E	Environment, too	/T:type	Set variable type
/M	Master environment	/U	User
/P	Pause or Prompt	/V	Volatile
/O	Don't overwrite	/X	Override VariableExclude
/Q	Don't echo /A result		

See also: ESET and UNSET.

Usage:

Every program and command inherits an environment, 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-RT itself uses several environment variables.

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, /S, /U, and /V).

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, 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 command. To remove variables from the environment, use UNSET, or type SET, followed by the variable name and an equal sign:

```
set mine=
```

The variable's *name* is limited to a maximum of 1024 characters.

Note: You cannot use SET to modify GOSUB variables.

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, /S, /U and /V 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, /S, /U or /V **will not set** the variable in the local environment unless you also use the /E 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 configuration option is set, TCC-RT will monitor the WM_SETTINGCHANGE message and update the environment from the User, Volatile, and System registry entries. The update is done whenever TCC-RT displays the prompt (to prevent the environment from changing in the middle of a command).

Array Variables

In addition to environment variables, SET is also used to set values for array variables. For example, to define a 5-row by 10-column array, you would first use SETARRAY:

```
setarray array1[5,10]
```

To set the array values (0-based), the syntax is:

```
set array1[a[,b[,c[,d]]]
```

For example:

```
set array1[0,0]=Bob
set array1[0,1]=Bob's Job
```

To expand the array variable:

```
echo Name is %array[0,0] and job is %array1[0,1]
```

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 to perform the same task; SET /A is included for compatibility with CMD. Unlike @EVAL, use of the >> or << shift operators in SET /A requires disabling their interpretation as redirection symbols by using SETDOS /X-6.

/B   Set a batch variable (%1 - %n). Only valid when TCC-RT is executing a batch file.

/D   Create/modify/delete a **default** variable in the registry (HKU\.DEFAULT\Environment).

/E   When used together with one of /D, /S, /U, or /V, set both the registry variable and the local environment variable.

/M   If a variable name is specified, change it to the original value when *TCC-RT* started. If no name is specified, revert the entire environment to the original environment when TCC-RT started.

/O   Don't overwrite existing values (only valid in combination with /R).

/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.

When used with a variable name and an optional prompt string, e.g. set /p *myvar*=Enter value, emulates the CMD behavior of allowing entry of a value for the variable. This is provided for compatibility reasons only. For more flexibility, use the TCC-RT ESET or INPUT commands.

/Q   Don't echo the result of /A when at the command line.

/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 for TCC-RT. 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. 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., trailing whitespace, redirection and pipe symbols (**<>|**), without the need for escaping the characters. If you reference the value of another variable in *value* (e.g., **x=%path;c:\jpsoft**), 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*.*

/RO   Create a read-only variable. Once you have set the variable, you cannot change it or UNSET it. Only environment variables can be read-only, not registry variables or array variables. A read-only variable will automatically be exported from an ENDLOCAL.

/S   Create/modify/delete a **system** variable in the registry (HKLM\System\CurrentControlSet\Control\Session Manager\Environment).

/T:type["regex"]   Set a variable type. If you try to set the variable to an incompatible type, SET will return an error. The supported types are:

int (or 1)	The variable can only contain 0-9
dec (or 2)	The variable can only contain 0-9, the decimal character, and the thousands separator
hex (or 3)	The variable can only contain 0-9 and A-F
bool (or 4)	The variable can only contain 0 or 1
alpha (or 5)	The variable can only contain A-Z and a-z
alnum (or 6)	The variable can only contain A-Z, a-z, and 0-9
regex (or 7)	The variable must match the specified regular expression

/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.

4.2.170 SETARRAY

Purpose: Define array variables

Format: SETARRAY [/F /T:*type* /R *filename* [/Z] *arrayname*] *name*[a[,b[,c[,d]]]] [...]

          a,b,c,d   Array dimensions

          /F(orce overwrite)   /T(ype)
          /R(ead)   /Z (resize)

Usage:

You can define up to 4-dimensional arrays. For example, to define a 5-row by 10-column array:

```
setarray array1[5,10]
```

The array elements are addressed in base 0, so to reference this array you would use 0-4 for the rows and 0-9 for the columns.

You can initialize arrays by appending [*value*] to the definition. For example, to initialize all of the array elements to 0:

```
setarray myarray[100] [0]
```

To set the variable elements, use the SET command.

If you don't enter any arguments, SETARRAY will display the currently defined arrays. If you don't enter any dimensions, SETARRAY will display the definition for that array. You can use wildcards in the array name.

See also @ARRAYINFO.

Options:

/F   Force overwrite of existing arrays (if any).

/R Read a file into a 1-dimensional array. SETARRAY will automatically determine the required size of the array. You can only combine the /R and /F options.

/T:type[:"regexpression"] Set a variable type. If you try to set the variable to an incompatible type, SET will return an error. The supported types are:

int (or 1)	The variable can only contain 0-9
dec (or 2)	The variable can only contain 0-9, the decimal character, and the thousands separator
hex (or 3)	The variable can only contain 0-9 and A-F
bool (or 4)	The variable can only contain 0 or 1
alpha (or 5)	The variable can only contain A-Z and a-z
alnum (or 6)	The variable can only contain A-Z, a-z, and 0-9
regex (or 7)	The variable must match the specified regular expression

Z Resize an existing array. For example:

setarray myarray[5,2]
...
setarray /z myarray[8,3]

4.2.171 SETERROR

Purpose: Set the ERRORLEVEL value

Format: SETERROR errorlevel

errorlevel New value for ERRORLEVEL

Usage:

SETERROR sets the value of the ERRORLEVEL internal variable and the last-error code in Windows to the specified value.

See also IF.

4.2.172 SETDOS

Purpose: Display or set the TCC-RT configuration

Format: SETDOS [/C? /D? /E? /Fn.n /G?? /I[+|-] command /M? /N? /P? /S?:? /V? /X[+|-]n]

/C(ompound)	/M(ode for editing)
/D(escriptions)	/N(o clobber)
/E(scape character)	/P(arameter character)
/F(ormat for @EVAL)	/S(hape of cursor)
/G (numeric separators)	/V(erbose)
/I(nternal)	/X (expansion, special characters)

See also: OPTION.

Usage:

SETDOS allows you to customize certain aspects of TCC-RT 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.

Inheritance

When a new instance of the command is started, it inherits the SETDOS characteristics set by the most recently started instance of TCC-RT.

Options:

/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 characters (**| > <**), or a space, tab, comma, or equal sign as the command separator. The command separator is saved by SETLOCAL and restored by ENDLOCAL. The following example changes the separator character to a tilde [**~**]:

    setdos /c~

/D   [Descriptions and Description Name] This option controls whether file processing commands like COPY, DEL, MOVE, and REN process file descriptions along with the files they belong to. **/D1** turns description processing on, which is the default. **/D0** turns description processing off.

You can also use **/D** to set the name of the hidden file in each directory that contains file descriptions. To do so, follow **/D** with the filename in quotes:

    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 will be passed unmodified to the command. The default escape character is a caret [**^**]. You cannot use any of the redirection characters (**| > <**) or a space, tab, comma, or equal sign as the escape character. The escape character is saved by SETLOCAL and restored by ENDLOCAL. Certain characters (**b**, **c**, **e**, **f**, **k**, **n**, **q**, **r**, **s**, and **t**) have special meanings when immediately preceded by the escape character.

/F   [@EVAL maximum and minimum] This option lets you set the default decimal display precision for the @EVAL 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 function if you want to set the display precision for a single computation.

/G [Decimal and thousands separator characters] This option sets the Decimal and Thousands separator characters. The format is **/Gxy** where "x" is the new decimal separator and "y" is the new thousands separator. Both characters must be included. The only valid settings are **/G.,** (period is the decimal separator, comma is the thousands separator); **/G,.** (the reverse); or **/G0** to remove any custom setting and use the default separators associated with your current country code (this is the default).

The decimal separator is used for @EVAL, numeric IF and IFF tests, version numbers, and other similar uses. The thousands separator is used for numeric output, and is skipped when performing calculations in @EVAL.

/I 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-RT 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**).

/N [NoClobber] This option controls output redirection. **/N0** means existing files will be overwritten by output redirection (with **>**) and that appending (with **>>**) does not require the file to exist already. This is the default. **/N1** means existing files may not be overwritten by output redirection, and that when appending the output file must exist. A **/N1** setting can be overridden with the [**!**] character.

/P [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 or alias. The default value is the dollar sign [**$**]. The parameter character is saved by SETLOCAL and restored by ENDLOCAL.

/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-RT 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.

/V [Batch Echo] This option controls the default for command echoing in batch files.

/V0 disables echoing of batch file commands unless ECHO is explicitly set ON.

/V1, the default setting, enables echoing of batch file commands unless ECHO is explicitly set OFF.

/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-RT 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
A	User-defined functions

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 command will save the current SETDOS /X values for ENDLOCAL to restore.

4.2.173 SETLOCAL

Purpose:	Save a copy of the current disk drive, directory, environment, alias and function lists, and special characters
Format:	SETLOCAL [GLOBALLISTS]
	GLOBALLISTS Global aliases / user variable functions

See also: ENDLOCAL.

Usage:

SETLOCAL can be used on the command line, in aliases, in library functions, and in batch files.

SETLOCAL will save :

- the default disk drive and directory
- the environment,
- the alias list
- the user-defined function list
- The directory stack (PUSHD)
- the special character set (command separator, escape character, parameter character, decimal separator, and thousands separator)
- the SETDOS /X setting
- the SETDOS /F 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 command, or just by exiting the batch file.

SETLOCAL does not save the command history or array variables.

If you have global aliases and/or functions, SETLOCAL will now copy them to a local list for the duration of the SETLOCAL. The matching ENDLOCAL will reset them to the global list. If you have both local and global aliases or functions defined, SETLOCAL will only save the local list (which will be restored by ENDLOCAL).

SETLOCAL supports the EnableExtensions, DisableExtensions, EnableDelayedExpansion, and DisableDelayedExpansion arguments from CMD. (Though they're not necessary, since TCC-RT either sets those by default or through the OPTION command.)

For example, this batch file fragment saves everything, removes all aliases so that 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 32 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 configuration option. Consequently changing aliases inside a SETLOCAL / ENDLOCAL pair affects the definition of aliases of other concurrently executing sessions of TCC-RT.

You can also use SETLOCAL and ENDLOCAL in an alias or at the command line. The maximum nesting level from a command line or alias is 32 levels. Unlike batch files, you are responsible for matching the SETLOCAL / ENDLOCAL calls from an alias or command line; TCC-RT will not perform an automatic ENDLOCAL.

An ENDLOCAL is performed automatically at the end of a batch file, or when returning from a "GOSUB filename". If you invoke one batch file from another without using CALL, the first batch file is terminated,

and an automatic ENDLOCAL is performed; the second batch file inherits the settings as they were prior to any SETLOCAL.

You can "export" modified variables from inside a SETLOCAL / ENDLOCAL block. See ENDLOCAL for details.

Options:

 GLOBALLISTS Prevent SETLOCAL from switching to local alias and user-defined variable function lists during the SETLOCAL duration.

4.2.174 SETP

 Purpose: Create, modify, or display an environment variable in another process

 Format: SETP *pid* [/P /R *filename*] var[=value]

    pid Process ID, or the window title, or the task name
    var The variable name to set. If you are displaying matching variables, the name can contain wildcards.
    value The value of the variable

    /P Pause after displaying each page
    /R Read variables and values from a file

See also UNSETP.

Usage:

SETP works by injecting a dll into the specified process and executing a command in that dll to set the environment variable. Depending on your Windows configuration, you may need to be running an elevated session for SETP to work.

Example:

Set a variable in the process whose window title begins with "ABC":

```
setp
```

4.2.175 SHIFT

 Purpose: Allows the use of more than 10 parameters in a batch file, or iterating through its parameters

 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, where it was used to access more than the CMD limit of 10 parameters. TCC-RT supports 8191 parameters (%0 to %8190), so you do not need to use SHIFT for batch files running exclusively under TCC-RT.

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, this form of the SHIFT command does not alter the contents or order of the parameters returned by `%*`. See [Batch File Parameters](#) for details.

Examples:

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). See [Batch File Parameters](#) for details.

4.2.176 SHORTCUT

Purpose: Create or display a shortcut

Format: [Creation mode](#)
             SHORTCUT *command args dir desc link mode* [*iconfile* [*iconoffset* [*hotkey*]]]

[Display mode](#)
SHORTCUT *link*

command	Command the shortcut executes
args	Command line parameters for command
dir	Starting directory
desc	Description
link	Filename of the .LNK 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-RT*** 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. SHORTCUT will not try to fully qualify the command name, startup directory, or link file name if they contain **%** characters. This allows you to embed variables that will be expanded by Windows.

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](#)) 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-RT 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.**

> **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; 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.

Example:

Create a shortcut for TCC-RT that starts in D:\TakeCommand28 in a normal window:

```
shortcut "C:\Program Files\JPSoft\TCMD28\tcc.exe" "" d:\TakeCommand28
"TCC command processor" tcc.lnk 1 "C:\Program
Files\JPSoft\TCMD28\tcc.exe"
 a single parameter (a link file name), SHORTCUT will display the values for that link.
```

4.2.177 SHRALIAS

Purpose: Retains global command history, directory history, alias and user function lists in memory when TCC-RT is not running

Format: SHRALIAS [/U]

/U(nload)

Usage:

When you close all TCC-RT 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-RT 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-RT is not running. SHRALIAS.EXE must be stored in the same directory as TCC-RT 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-RT 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-RT* session which exits after starting SHRALIAS, the TCC-RT session may terminate and discard the shared lists before SHRALIAS can attach to them. In this case SHRALIAS.EXE will not be loaded. If you experience this problem, add a short delay with the DELAY command after SHRALIAS is loaded and before your session exits.

SHRALIAS will not work in detached sessions (i.e., those started with DETACH, 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 Functions, and Local and Global Aliases.

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-RT exits unless SHRALIAS is loaded again before that time.

4.2.178 SMPP

Purpose: Send simple text (**SMS**) messages, typically to text-enabled cellular phones and similar devices

Format: SMPP [/IPv6] *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
/IPv6	Use IPv6 instead of IPv4.

See also: SENDMAIL, SNPP.

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.

4.2.179 SNPP

Purpose: Send messages to alphanumeric pagers

Format   SNPP [/IPv6] *server pagerid message*

server	The SNPP server name
pagerid	The ID of the pager to receive the message
message	The message to send
/IPv6	Use IPv6 instead of IPv4.

See also: SENDMAIL, SMPP.

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.

4.2.180 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:

Trap Name	OID
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

4.2.181 SPONGE

Purpose: Read standard input and write it to a file.

Format: SPONGE [/A] *outputfilename*

/A(ppend)

Usage:

Unlike output redirection, SPONGE reads all its input before opening the output file. This allows constructing pipes that read from and write to the same file. SPONGE reads standard input into a memory buffer, so piping extremely large amounts of data (i.e., multiple gigabyte) is not recommended.

Options:

/A   **A**ppend output to *outputfilename*. The default is to overwrite *outputfilename*.

4.2.182 SSHEXEC

Purpose: Connect to server using SSH and start default shell

Format: SSHEXEC [/A /F *filename* /G*n* /H *fwhost* /IPV6 /R *port* /S /T *type* /U *user* /P *password*] *host* /L *name*[:*password*] "*command* ..."

[/A (firewall autodetect)](#)   [/P (firewall password)](#)
[/F (file for stdin)](#)   [/R(emote port)](#)
[/G (log level)](#)   [/S(tatus messages)](#)
[/H (firewall host)](#)   [/T (firewall type)](#)
[/IPv6](#)   [/U (Firewall user name)](#)
[/L (user:password)](#)

host - Host name
command - Command to pass to the host's default shell

Usage:

The SSHEXEC command establishes a Secure Shell (SSH) connection to a server and starts up the user's default shell. Press Ctrl-C to disconnect from the other system.

If you don't specify a user name, SSHEXEC will use the current user name. You can provide a password on the command line by appending it to the user name (i.e., "User:Password"). If you don't provide a password, SSHEXEC will prompt for it.

If you want to do redirection on the remote system, enclose the command argument list in double quotes. The double quotes will be removed before passing the commands to the remote system.

SSHEXEC will display the host name & user name and prompt for a line of input, then send it to the host shell and return to the prompt to wait for the next line. SSHEXEC will display any output sent by the host to STDOUT and STDERR. When you type "exit" at the prompt, or the host disconnects SSHEXEC will exit.

Options:

/A   Automatically detect and use firewall system settings, if available

/F   Send the contents of a file as the stdin input to the SSH server.

/G*n*   The level of detail that is logged (for debugging connection issues). The possible values are:

       0   No messages are logged
       1   Informational events such as SSH handshake messages are logged
       2   Detailed data such as individual packet information is logged
       3   Debug data including all relevant sent and received bytes are logged

/H   Firewall host name

/IPV6 Use IPv6 instead of IPv4

/L   User name (ID).

/P   Firewall user password

/R   Remote port number

/S   Display SSH status messages (for debugging connection issues)

/T*n*   Firewall type, where ***n*** is:

      1   Connect through a tunneling proxy. The firewall port is set to 80.
      2   Connect through a SOCKS4 proxy. The firewall port is set to 1080.
      3   Connect through a SOCKS5 proxy. The firewall port is set to 1080.
    10   Connect through a SOCKS4A proxy. The firewall port is set to 1080.

/U   Firewall user name

4.2.183 START

Purpose: Start a program in another session or window

Format: START ["*title*"]
[/= /AFFINITY=
n /ABOVENORMAL /BELOWNORMAL /BREAKAWAY /COLOR=*BF* /DESKTOP=*name*
/ELEVATED /Env=*file* /FEEDBACK=off|
on /HIGH /LOW /JOB=*jobname* /NOPINNING /NORMAL /PARENTAFFINITY /REALTI
ME /VDESKTOP=*id* /B /C /K /D*path* /I /INV /MAX /MIN /NODE
n /POS=*x,y,width,height* /L /LA /LD /LF /LH /MONITOR=n /RUNAS *user
password* /SIZE=rows,cols /TAB /TABNA /UNELEVATED /WAIT /WIN /PGM]
"*progname*" [*command*]

title	Title to appear on title bar
path	Startup directory
progname	Program name (not the session name)
command	Command to be executed by ***progname***

/ABOVENORMAL	Priority	/LF	Local functions
/AFFINITY	Multiple CPUs	/LH	Local history list
/B	No new console	/LOW	Priority
/BELOWNORMAL	Priority	/MAX	Maximized window
/BREAKAWAY	Break away from job	/MIN	Minimized window
/C	Close when done	/MONITOR	Monitor to use
/COLOR	Default console colors	/NODE	NUMA node
/D	Startup directory	/NOPINNING	Don't pin to taskbar
/DESKTOP	Start desktop	/NORMAL	Priority
/ELEVATED	Start as admin	/PARENTAFFINITY	Inherit parent's affinity
/Env	Custom environment	/PGM	Program name
/FEEDBACK	Cursor feedback mode	/POS	Position of window
/HIGH	Priority	/REALTIME	Priority
/I	Inherit environment	/RUNAS	Run as other user
/Idle	Wait for input idle	/SIZE	Screen buffer size
/INV	Invisible window	/TAB	Start in *Take Command* tab window

/JOB	Start process in job	/TABNA	Start in inactive **Take Command** tab
/K	Keep when done	/UNELEVATED	Start app unelevated
/L	Local lists	/VDESKTOP	Start on the specified virtual desktop
/LA	Local aliases	/WAIT	For session to finish
/LD	Local directory history	/WIN	Windowed session

See also: DETACH.

Usage:

START is used to begin a new session, and optionally run a program in that session. If you use START with no parameters, it will begin a new TCC-RT 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.

If *title* is included, it will appear on the task list and **Alt-Tab** displays instead of the program name. *Title* must be enclosed in double quotes, and cannot exceed 127 characters. *Title* will be ignored if you also specify /ELEVATED.

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 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 (a sequence of 0-9 digits) 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: /MAX, /MIN, and /POS allow you to start a character-mode windowed session in a maximized window, a minimized window, or a window with a specified position and size, respectively. /INV starts an invisible window. /B starts the program in the current console window. The default is /WIN, which permits Windows to choose the position and size of the non-maximized window. If you start a graphics mode program, only /MAX and /POS are effective, and the position and size information associated with /POS 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 command after the window has been opened.

Session priority: The options /ABOVENORMAL, /BELOWNORMAL, /HIGH, /LOW, /NORMAL and /REALTIME allow you to select the new session's priority.

Program controls.

If *progname* is in the "App Paths" registry (either HKCU or HKLM), its associated "Path" value (if it exists) is inserted into the beginning of the PATH in the environment inherited by the program.

If *progname* is the name of a directory instead of an executable program, TCC-RT will start your default Windows shell (usually Windows Explorer) in the specified directory.

Progname inherits the environment as it exists when START is executed, unless /I is used to select the default environment.

If *progname* specifies **TCC.EXE**, the options /L, /LA, /LD, /LF and /LH provide control over the use of local or global lists. See details below.

The initial directory for *progname* is the current default directory, unless otherwise specified using the /D option.

When *command* is finished, /C closes the session (the default for Windows sessions), while /K 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 internal variable.

Options:

/=	Display the START command dialog to help you set the filename and command line options. The /= option can be anywhere on the line; additional options will set the appropriate fields in the command dialog.
/ABOVENORMAL	Set the priority above normal.
/AFFINITY=n	On multiple processor machines, set the processor affinity for this process. /AFFINITY takes a hex argument for the processor mask -- i.e., to set the affinity for cpu's 1 and 3, set /AFFINITY=5. If you also specify a NUMA node (see /NODE), the process is restricted to running on those processors in common between the specified affinity mask and the NUMA node. If no processors are in common, the process is restricted to running on the specified NUMA node.
/B	The program is started without creating a new window or console, i.e. in the TCC-RT window. Normally, the application is started in its own window. For compatibility with CMD, /B also disables Ctrl-C processing for the program.
/BELOWNORMAL	Set the priority below normal.

/BREAKAWAY	The child process is not associated with the TCC-RT job (see JOBS). This requires that TCC-RT is running in a job with the breakaway option enabled..
/C	Start the program in a new TCC-RT window and close the TCC-RT window when the application ends.
/COLOR=BF	Set the default color for the new console window. **B** is the background color (hex 0-F) and **F** is the foreground color (hex 0-F).
/D	Specifies the startup directory. Include the directory name immediately after the /D, with no intervening spaces or punctuation.
/DESKTOP	Specify the window station and desktop where the app should be started. If you don't enter a backslash (\), the argument is assumed to be the desktop on **WINSTA0** where you want the app to start.
/ELEVATED	Start the program elevated, with full admin privileges. /ELEVATED cannot be used in combination with /RUNAS. If you specify /ELEVATED, any **title** on the command line will be ignored (this is a Windows limitation).

/Env="_filename_**"**	Creates a new environment for the process using the contents of _filename._ The format of _filename_ is:
	
	    var1=value1
	    var2=value2
	
	    _..._
	
	Because some Windows API calls will fail if there is no "SystemRoot" variable in the environment, TCC-RT will add the existing _SystemRoot_ value to the new environment if wasn't specified in _filename_.

/FEEDBACK=_on|off_	**ON -** The cursor is in feedback mode for two seconds after the process is started, and the "Working in Background" cursor is displayed. If during those two seconds the process makes the first GUI call, the system gives five more seconds to the process. If during those five seconds the process shows a window, the system gives five more seconds to the process to finish drawing the window. The system turns the feedback cursor off after the first call to GetMessage, regardless of whether the process is drawing. **OFF** - The feedback cursor is forced off while the process is starting (the normal select cursor is displayed).
/HIGH	Start the window at high priority.
/I	Inherit the default (startup) environment, rather than the current environment.

/Idle=*n*	Waits until the started process to finish processing its initial input and is waiting for user input, or until the timeout period has elapsed. *n* specifies the timeout period in milliseconds.
/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.
/JOB=*jobname*	Start the new process in the specified job (see JOBS). Cannot be used with /RUNAS.
/K	Start the program in a new TCC-RT window and keep the TCC-RT window open when the program ends. (Use the EXIT command to close the TCC-RT window.)
/L	Start TCC-RT with local alias, function, history and directory history lists. This option is equivalent to specifying all of /LA, /LD, /LF, and /LH (below).
/LA	Start TCC-RT with a local alias list. See ALIAS for information on local and global alias lists.
/LD	Start TCCwith a local directory history list. See Local and Global History Lists for information on local and global directory history lists.
/LF	Start TCC-RT with a local function list. See FUNCTION for information on local and global function lists.
/LH	Start TCC-RT with a local history list. See Local and Global History Lists 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=n	Start the program on the specified monitor (1 to n). This will only work with apps that do not try to position their window at startup, and you cannot combine this switch with /POS.
/NODE n	Start the program using the specified NUMA node (*n* is a decimal integer). See also /AFFINITY.
/NOPINNING	Any windows created by the new process cannot be pinned on the taskbar.
/NORMAL	Start the window at normal priority.
/PARENTAFFINITY	The process inherits its parent's affinity.
/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=left,top,width,height	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.) /RUNAS cannot be used in combination with /ELEVATED.

If the user name begins with ".\" (without the quotes) , TCC-RT will substitute the computer name for the ".".

/SIZE=rows,columns	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-RT window, and activate the new window. The command will usually be a Windows console mode application, but *Take Command* can also run many simple GUI applications in a tab window (provided the application does not have multiple parent windows).
/TABNA	Start the command in a new TCC-RT window, but don't activate the new window.
/UNELEVATED	Start the program in an unelevated session. (Only necessary if TCC-RT is running in an elevated session and you want to start a process unelevated.)
/VDESKTOP=*id*	Start the app on another virtual desktop. *id* can be either a desktop number (1-n), the GUID for that desktop, or the desktop name. See VDESKTOP for more details. Note that Windows doesn't have an API to actually start on another desktop, so TCC-RT starts it on the local desktop and them immediately moves it -- you'll see a flash when the window starts and then disappears.
/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 /B.

4.2.184 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*]
CASEALL
   [*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 and IFF.

Usage:

SWITCH can only be used in batch files. It allows you to select a command or group of commands to execute based on the possible values of a variable or a combination of variables and text.

The SWITCH command is always followed by an **expression** created from environment variables, internal variables, variable functions, and text strings, and then by a sequence of CASE statements matching the possible **values** of **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-RT 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 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. The **value** in a CASE statement can be literals, or variables or functions (which will be expanded prior to the comparison with the SWITCH expression).

CASE statements can include wildcards and regular expressions.

The optional CASEALL statement should follow all of the CASE statements but precede DEFAULT. If any preceding CASE block was executed, CASEALL will also be executed; otherwise it is 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](#) 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
```

4.2.185 SYNC

Purpose: Synchronize two directories

Format: SYNC [/= /A:... /C /D /DD /E /F /G /J /K /L /M /N[enrst] /O /O:[-]
acdeginorstuz /P /Q /R /S[[+]n] /T /V /W /X /Y /Z] *dir1 dir2*

> *dir1*   First directory (and source for a /W)
> *dir2*   Second directory (and target for a /W)

/A:...	Attribute switch	/O	Only if no target file
/C.	Changed source files	/O:...	Sort order
/D	Copy encrypted files	/P	Prompt
/DD	Remove empty subdirectories	/Q	Quiet
/E	No error messages	/R	Replace
/F	No empty subdirectories	/S	Subdirectories included
/G	Display percentage completed	/T	Totals
/H	H(idden included)	/V	Verify
/I"text"	Match description	/W	Delete non-matching target
/J	Restartable copy	/Wait	Wait between block copy
/K	Keep RDONLY attribute	/X	Clear archive bit
/L	ASCII-mode FTP transfer	/Y	Suppress prompt
/M	Modified files (not Archived)	/Z	Overwrite read-only
/N	Disable		

See also: COPY and MOVE.

File Selection

Supports extended wildcards and ranges.

Internet: Can be used with FTP servers.

Usage:

SYNC will synchronize two directories, copying the updated files from each directory to the other. If you don't specify any arguments, SYNC will display its command dialog.

SYNC sets three internal variables:

%_sync_dirs	The number of directories created
%_sync_files	The number of files copied
%_sync_errors	The number of errors

Example:

Synchronize the directories C:\MyData and D:\MyData, including subdirectories (but not empty subdirectories), and delete any files in D:\MyData that don't exist in C:\MyData:

```
sync /S /F /W C:\MyData D:\MyData
```

Options:

/=　　Display the SYNC command dialog to help you set the directory and command line options. The /= option can be anywhere on the line; additional options will set the appropriate fields in the command dialog.

/A:...　　Select only those files that have the specified attribute(s) set. See Attribute Switches 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. Hidden or system files selected by this option overwrite hidden or system files in the target directory.

　　You can specify **/A:=** to display a dialog to help you set individual attributes.

/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.

/D　　Force copy of an encrypted file even when the target will be decrypted.

/DD　　When used with **/S**, SYNC will delete any empty target subdirectories.

/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 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 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 switch, or use ATTRIB.

/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.

A **/N** with one or more 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)
n	Don't update the file descriptions
r	A SYNC /W will delete to the recycle bin (unless the file matches the RECYCLEEXCLUDE environment variable).
s	Don't display the summary
t	Don't update the CD / CDD database *(JPSTREE.IDX)*

/O   Only if no target file.

/O:...   Sort the files before processing.

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, instead of numerically when numeric substrings are included in the name or extension.
c	Sort by compression ratio
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 description
o	Sort by owner
r	Reverse the sort order for all options
s	Sort by size
t	Same as **d**
u	Unsorted
z	Same as **s**

/P   Ask the user to confirm each copy and delete. Your options at the prompt are explained in detail under Page and File Prompts. See also: the /Q option below.

/Q   Don't display filenames, percentage copied, total number of files copied, etc... 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 the user before overwriting an existing file. Your options at the prompt are explained in detail under Page and File Prompts.

/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 search 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.

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.

If you specify a **+** followed by a number after the /S, SYNC will not sync any files until it gets to that depth in the subdirectory tree. For example, if you have a directory tree \a\b\c\d\e, /S+2 will not sync anything in \a or \a\b.

/T	Turns off the display of filenames, like /Q, 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**.
/Wait=n	Pause for *n* milliseconds between each block copied from the source to the target file. This is useful for slow networks and very large file copies; it prevents SYNC from monopolizing all of the network I/O.
/X	Clear the archive attribute from the source file after a successful copy.
/Y	If you have the "COPY Prompt on Overwrite" option set, you can suppress the prompt with /Y.
/Z	Overwrite **destination** files regardless of their attributes. Without this option, SYNC 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 or /A: is used to select it.

4.2.186 TAIL

Purpose:	Display the end of the specified file(s)
Format:	TAIL [*range* ... [/I"*text*"]] [/A:[attrlist] /B /Cnn /F /N+x /N[ ]n /O:[-]acdeginorstuz /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)
/B(ell)	/O:... (Order)
/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 "https://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 (*.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.

TAIL sets two internal variables:

%_tail_files	The number of files displayed
%_tail_errors	The number of errors

TAIL will recognize Unicode (UTF-16) files based on either a BOM or specific UTF-16 sequences at the beginning of the file. TAIL will recognize UTF-8 files based on either a BOM or UTF-8 extended characters within the first 2K of the file.

- ***FTP Usage***

TAIL can also display files on FTP servers. For example:

```
tail "ftp://ftp.microsoft.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. 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:

`tail /n 5 xxx`	displays the last 5 lines of file *xxx*
`tail /n+20 /n 999999 xxx`	skip 20 lines, then display 999999 lines of *xxx*
`tail /n+1001 /n 1 xxx`	skip 1001 lines, then display 1 line of *xxx*
`set x=%@execstr[tail /n+1001 /n 1 xxx]`	sets **x** to the contents of line 1002 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 for details.

　　　　　You can specify **/A:=** to display a dialog to help you set individual attributes.

/B　　　Ignore bell (ASCII 7) characters.

/C*nn[b|k|m]*
　　　　　Display **nn** bytes, 512-byte **b**locks, **k**ilobytes, or **m**egabytes.

/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. 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 is also specified.

/N+x　Skip **x** lines from the beginning of the file, then 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.

/O:...　Sort the files before processing.

　　　　　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, instead of numerically when numeric substrings are included in the name or extension.

`c`	Sort by compression ratio
`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 description
`o`	Sort by owner
`r`	Reverse the sort order for all options
`s`	Sort by size
`t`	Same as `d`
`u`	Unsorted
`z`	Same as `s`

/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.

4.2.187 TAR

Purpose: Add, update, or delete files in a .tar archive

Format: TAR [/A:[[-][+]rhsdaecjot] /A /C /D /F /G /M /O:[-]adegnrstu /Q /R /TEST /U /V] *tararchive* [*@file*] *file*...

        tararchive   The tar file to work with
        file   The files(s) to be added to the tar archive

/A:... (attribute switch)	/O:... (sort order)
/A(dd)	/P(rogress)
/C(ontents)	/R(ecurse)
/D(elete)	/TEST
/F(reshen)	/U(pdate)
/G(zip)	/V(iew)
/M(ove)	

        See also UNTAR.

File Selection

Supports attribute switches, extended wildcards, ranges, multiple file names, and include lists.

Usage:

TAR is compatible with archives created by the Linux / UNIX tar utility. Unless you use the /G option, the tar file will be uncompressed.

You can specify a pathname for tar*archive*. If you don't provide an extension, and the filename as entered doesn't exist, TAR adds ".tar". If you don't specify an operation, TAR will default to Add.

TAR supports connected web folders. If an HTML file (i.e., with an .htm or .html extension) is compressed, TAR 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 be added the TAR archive.

TAR sets two internal variables:

| %_tar_files | The number of files archived |
| %_tar_errors | The number of errors |

Option:

/A:... Select only those files that have the specified attribute(s) set. See Attribute Switches for information on the attributes which can follow /A:. Do not use /A: with ***@file*** lists. See @file lists for details.

You can specify **/A:=** to display a dialog to help you set individual attributes.

/A Add the specified file(s) to the tar file. (This is the default.)

/C Display (on standard output) the contents of a file in the tar archive.

/D Delete the specified file(s) from the tar file.

/F Update only those files that currently exist in the tar file, and which are older than the files on disk.

/G When all the files have been added to the archive, compress the entire archive using gzip compression and create a .tar.gz archive.

/M Delete the files from the disk after adding them to the tar file.

/O:... Sort the files before processing.

You may use any combination of the sorting options below. If multiple options are used, the files 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, instead of numerically when numeric substrings are included in the name or extension.
`d`	Sort by date and time (oldest first); also see **/T:acw**
`e`	Sort by extension
`g`	Group subdirectories first, then files
`r`	Reverse the sort order for all options
`s`	Sort by size
`t`	Same as `d`
`u`	Unsorted

/P Display the progress (0 - 100%) for each file as it is archived.

/Q Don't display the files being archived.

/R   If the argument is a subdirectory, copy all of the files in that subdirectory and all of its subdirectories to the tar archive.

/TEST   Test the integrity of the TAR file (header and contents). Any errors will be displayed on STDERR.

/U   Update files which either don't exist in the tar, or which are older than the files on disk.

/V   View the list of files in the tar file (date, time, size, and filename).

4.2.188 TASKBAR

Purpose: Call Windows Taskbar functions

Format: TASKBAR *command*

Usage:

TASKBAR calls the Windows Taskbar to display dialogs or to manipulate the top level windows.

Options:

Cascade	Cascade all top level windows.
Computers	Display the Find Computers dialog (requires Active Directory Domain Services)
Control	Display the Control panel.
Customize	Display the Customize Taskbar dialog.
Date	Display the Date and Time dialog.
Desktop	Show the Windows desktop.
Help	Display the Help and Support Center dialog.
HTile	Horizontally tile all top level windows.
Lock	Toggle the taskbar lock.
Logoff	Display the Logoff dialog.
Min	Minimize all windows.
Max	Maximize all windows.
Printers	Display the Printers and Faxes dialog.
Properties	Display the Taskbar Properties dialog.
Run	Display the Run dialog.
Search	Display the Search dialog.
Shutdown	Display the Shut Down Computer dialog.

Start	Display the Start Menu.
Task	Display the Windows Task Manager dialog.
VTile	Vertically tile all top level windows.

4.2.189 TASKDIALOG

Purpose: Display a Windows Task Dialog

Format: TASKDIALOG [/A[F]
[X]"*Details*" /AC"*text*" /AE"*text*" /B"text" /C /DB:n /DR:n /E /F[EISW]"text" /H /I /L /N /P=*x,
y* /R"*text*" /S /T:n /V"*text*" /W /X] *buttontype* "*title*" "*instruction*" [*text*]

buttontype	One or more of **OK**, **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

/A (details)	/L(inks)
/B(utton text)	/N (links)
/C(heckbox)	/P(osition)
/DB (default button)	/R(adio buttons)
/DR (default radio button)	/S(top icon)
/E (shield)	/T(imeout)
/F(ooter)	/V(erification)
/H(yperlinks)	/W(arning icon)
/I(nformation icon)	/X (closable)

See also: INKEY, INPUT, MSGBOX and QUERYBOX.

Usage:

The button the user chooses is indicated using the internal variable % ?. Be sure to save the return value in another variable or test it immediately; because the value of % ? changes with every internal command. The following list shows the value returned for each button:

response	% ?
Yes or OK	10
No	11
Cancel or Close	12
Retry	13

If there is an error in the TASKDIALOG command itself, % ? will be set to 2.

Since TASKDIALOG doesn't write to standard output, it disables redirection allow you to enter the redirection characters (< and >) in your prompt text. If you want to use pipe characters or command separators, you will need to escape or quote them.

TASKDIALOG creates a popup dialog box. If you prefer to retrieve input from the command line, see the INKEY and INPUT commands.

You can copy the text in a TASKDIALOG window to the clipboard by entering Ctrl-C when the TASKDIALOG window has the keyboard focus.

TASKDIALOG can set three internal variables:

> _taskdialog_button - the button pressed to exit TASKDIALOG
> _taskdialog_radio - the selected radio button (if any) in TASKDIALOG
> _taskdialog_verify - returns 1 if the verify button was checked

Example:

Display a Yes / No message box and take action depending on the result:

```
taskdialog yes no "Copy" "Copy all files to A:?"
if %_? == 10 copy * a:
```

Options:

/A"*details***"**	TASKDIALOG will show a button that you can click to expand the dialog and view the text specified in "*Details*".
/AC"*text***"**	The button text for collapsing the expandable information.
/AE"*text***"**	The button text for expanding the expandable information.
/AF"*details***"**	Like /A, but TASKDIALOG will show the details at the bottom of the dialog's footer area instead of immediately after the contents.
/AX"*details***"**	Like /A, but TASKDIALOG will show the expanded details by default.
/B"*text***"**	Text to use for custom buttons. If you specify one or more /B arguments, TASKDIALOG will not display any of the default buttons. TASKDIALOG will return the button ID of the button pushed in the command variable %_taskdialog_button. TASKDIALOG will number the custom button ID's beginning at 1000. The maximum number of custom buttons is 10.
/C	Check the verification checkbox at TASKDIALOG startup. (The checkbox defaults to unchecked.)
/DB:*xx*	Default button. This can either be a number (1000-n for custom buttons, or a defined button type:

> OK
> Yes
> No
> Cancel
> Retry
> Close

/DR:*n*	The default radio button (2000 - n, only valid when used with /R).
/E	Display the Windows security shield icon.
/F"*text***"**	Display footer text with an optional icon:

E	security shield
I	information
S	error
W	warning

/H Enable hyperlinks embedded in the additional info (/A) text, the footer (/F) text, and the main instruction text. Hyperlinks are created with an <a> HTML tab. For example:

   /A"This is a hyperlink: Full details about Take Command 28"

/I Display an icon consisting of a lower case "i" in a circle in the message box.

/L Convert the buttons defined by /B into command links. A command link is a bigger button that has an icon and optionally a second smaller line of text. (To display a second line, append a ^n to the /B argument, followed by the text for the second line.)

/N Custom buttons (see /B and /R) are to be displayed as command links instead of push or radio buttons.

/P Display the task dialog at the specified screen coordinates. If /P is not specified, TCC-RT will center the dialog.

/R"*text*" Display radio buttons. The selected button will be returned in the command variable %_taskdialog_radio. TASKDIALOG will number the custom radio button ID's beginning at 2000. The maximum number of radio buttons is 10.

/S Display a stop sign icon in the message box.

/T:*n* Timeout after *n* seconds. If TASKDIALOG times out, it will return the Cancel / Close button value (12).

/V"*text*" Display a verification checkbox. If the box is checked, the command variable %_taskdialog_verify will be set when TASKDIALOG exits.

/W Display an exclamation point icon in the message box.

/X The task dialog can be closed using Alt-F4, Escape, and the title bar's close button even if no cancel button is specified.

4.2.190 TASKEND

Purpose: End the specified process

Format: TASKEND [/F /Ne /R] *pid* | *name* | "*title*"

pid	The process ID
name	The process name
title	Window title

[/F(orce)](#)   [/R(emove process tree)](#)
[/Ne (no errors)](#)

See also: TASKLIST, _PID, _DETACHPID, _WINTITLE

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 (either decimal or hex with a leading 0x), 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 the Windows DEBUG privilege, so (depending on the Windows version and the process you are trying to end) it may not work in a limited user account.

Example:

Force the process whose window title is "BadApp" to exit:

```
taskend /F "BadApp"
```

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.
/Ne	Don't display errors.
/R	Delete the process tree (the specified process and all of its child processes).

4.2.191 TASKLIST

Purpose:	Display a list of active processes
Format:	TASKLIST [/C /D /H /I /L /M /N /O /P[*n*] /R /T /U /U"*owner*" /X /Z] [*name*]

         name   Process name or window title

/C (Priority)	/P(ause)
/D (show modules)	/R (process tree)
/H (threads)	/T(ime)
/I(ntegrity)	/U(ser names)
/L (Startup command)	/U"owner"
/M(emory)	/X (hex)

/N (class names)   /Z (parent PIDs)
/O(rder by PID)

See also: TASKEND.

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. You can also optionally display the process priority, the modules (dll's) loaded by that process, the startup command line, the memory usage, the class name of the main window of the process, and the cpu usage.

TASKLIST displays a footer with the total number of processes, and optionally the total number of threads (if you specified /H).

If *name* begins with a **=**, it is assumed to be a process ID instead of a process name or window title. The Process ID can either be decimal or hex (with a leading 0x).

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.

Options:

/C	Display the current priority class for each process. The priority values are (in hex):

8000	Above normal
4000	Below normal
100	Realtime
80	High
40	Idle
20	Normal

/D	Display the loaded modules for each process.
/H	Display the threads for each process (thread ID and priority). If /X is also specified, the TID and priority will be displayed in hexadecimal format.
/I	Display the integrity (low, medium, high, system) for the code integrity and the resource integrity.
/L	Display the startup command line for each process.
/M	Display the memory usage for each process.
/N	Display the class name for the main window of each process.
/Nf	Don't display the TASKLIST footer.
/O	Sort the output by Process ID (PID).

/P[*n*]	Wait for a key to be pressed after each screen page before continuing the display. The /P option has an optional argument *n* that specifies the number of seconds to wait for a keystroke before continuing.
/R	Display the process tree (the specified process and all of its child processes).
/T	Display the system and user cpu usage for each process.
/U	Display the user name for each process (system processes return an empty string).
/U"owner"	Display only the processes for the specified user.
/X	Display the PIDs in hex.
/Z	Display the parent PIDs in the second column.

4.2.192 TEE

Purpose: Copy standard input to both standard output and a file

Format: TEE [/A /D*n* /F"*format*" /R /T] *file*...

   file   One or more files that will receive the "tee-d" output.

   [/A(ppend)](#)   [/R (STDERR)](#)
   [/D(ate)](#)   [/T(ime)](#)
   [/F"..." (format)](#)

See also: [Y](#), [piping](#) and [redirection](#).

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.

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](#) for more information on pipes.

Example:

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
```

Options:

/A	Append to the file(s) rather than overwriting them.
/D	Prefix each line with the current date (in yyyy-mm-dd format).
/F"..."	The *format* string. See @DATEFMT for details on *format* arguments.
/R	Redirect to STDERR instead of STDOUT.
/T	Prefix each line with the current time (in hh:mm:ss.ms format).

4.2.193 TEXT

Purpose: Display a block of text in a batch file

Format: TEXT
   .
   .
   .
   ENDTEXT

See also: ECHO, ECHOS, SCREEN, SCRPUT, and VSCRPUT.

Usage:

TEXT can only be used in batch files. Both TEXT and ENDTEXT must be entered as the only commands on their respective lines, and cannot be included in a command group.

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 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 or the COLOR command to set the default screen colors before executing TEXT.

Redirecting TEXT output

To redirect or pipe the entire block of text, use redirection or piping on the TEXT command itself as shown in the examples 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 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
```

4.2.194 THREAD

Purpose:	Execute a command in a separate thread.
Format:	THREAD *command [args]*

Usage:

It is the user's responsibility to ensure that there are no I/O or file system conflicts when running multiple THREAD commands and/or running THREAD simultaneously with commands in the primary TCC-RT thread.

THREAD will set the internal variable _thread_result to the return value of *command*.

4.2.195 TIME

Purpose: Display or set the system time

Format: TIME [/S [*server*] /T /U *"format"*] [*hh*[:*mm:ss*]]] [AM | PM]

     hh   The hour (0 - 23)
     mm   The minute (0 - 59)
     ss   The second (0 - 59)

     "..."   Date display format

     /S(erver time)   /U (UTC time)
     /T (Display only)

See also: DATE.

Usage:

If you don't enter any parameters, TIME will display the current system time and prompt you for a new time. Press Enter if you don't wish to change the time; otherwise, enter the new time:

```
[c:\] time
Wed  Mar 17, 2019  9:30:06
Enter new time (hh:mm:ss):
```

TIME defaults to 24-hour format, but you can optionally enter the time in 12-hour format by appending **a**, **am**, **p**, or **pm** to the time you enter. For example, to enter the time as 9:30 am:

```
time 9:30 am
```

The day of week and the month are translated into your local language (English, French, German, Italian, Russian, and Spanish).

Options:

    "..."   Custom date / time format to use when displaying the current time. The formatting characters are the same as used by the @DATEFMT function.

    /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 configuration option (the default is `clock.psu.edu`). Changing the time requires TCC-RT to be running in an elevated session.

/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.

/U   Display or enter the UTC time.

4.2.196 TIMER

Purpose: TIMER is a system stopwatch

Format: TIMER [/1 /2 /3 /4 /5 /6 /7 /8 /9 /10 /C /L /M /N /Q /S] [ON | OFF] [*command*]

ON   Force the stopwatch to reset and start
OFF   Force the stopwatch to stop
command   Time the specified command

/1   stopwatch #1 (default)   /C   clear on ^C
/2   stopwatch #2   /L   milliseconds
/3   stopwatch #3   /M   microseconds
/4   stopwatch #4   /N   nanoseconds
/5   stopwatch #5   /Q   quiet
/6   stopwatch #6   /S   split
/7   stopwatch #7
/8   stopwatch #8
/9   stopwatch #9
/10 stopwatch #10

Usage:

The TIMER command accepts its parameters in any order, and acts on the specified one of ten 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.

TIMER uses the Windows performance counters for greater accuracy. The default TIMER resolution is in milliseconds (.001 seconds).

The switch arguments (/1 - /10, /Q, and /S) must appear before any other arguments on the TIMER command line.

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 ten stopwatches available (1 - 10) 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
```

You can optionally specify a command for TIMER to run. This is the equivalent of "timer on & command & timer off". For example:

```
timer dir c:\ /s
```

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[] function.

Options:

/1   Use timer #1 (the default).

/2   Use timer #2.

/3   Use timer #3.

/4   Use timer #4.

/5   Use timer #5.

/6   Use timer #6.

/7   Use timer #7.

/8   Use timer #8.

/9   Use timer #9.

/10 Use timer #10.

/C Turn off the timer when a ^C is detected.

/L When used with /S (split time) or TIMER OFF, display the result in the number of milliseconds.

/M When used with /S (split time) or TIMER OFF, display the result in the number of microseconds.

/N When used with /S (split time) or TIMER OFF, display the result in the number of nanoseconds.

/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.

4.2.197 TITLE

Purpose: Change the window title

Format: TITLE [/P] *title*

[/P(rompt characters)](#)

title The new window title.

See also: the [ACTIVATE](#) and [WINDOW](#) commands.

Usage:

TITLE changes the text that appears in the caption bar at the top of the TCC-RT 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.

If you do not specify a new title, TITLE will display the existing console title.

Example:

To change the title of the current window to "Title Test":

```
title Title Test
```

Options:

    /P   Support the special characters in PROMPT.

4.2.198 TMP

Purpose: TMP displays or modifies the 10 temporary pseudo character devices available in TCC-RT (TMP0: - TMP9:)

Format: TMP [/C *tmpn:*] [/S *clipn: text*] [/Z]

    /C - Clear
    /S - Set TMP text
    /Z - Clear all

See also CLIP.

Usage:

TCC-RT supports multiple temporary character devices. They are numbered from TMP0: - TMP9:. TMP0: - TMP9: are only accessible to TCC-RT internal commands and variable functions.

TMP0: - TMP9: are similar to CLIP-: - CLIP9:, but are a little faster because they always work in UTF16 (so they don't translate to/from ANSI), and they don't need to access the Windows Clipboard (for CLIP0:). They also do not rotate like CLIPn: when something is pasted to the Windows Clipboard. Like CLIP*n:,* TMP*n:* values are local to the current session of TCC-RT.

The /C and /S options accept either TMPn: (i.e., TMP0: to TMP9:), or just the digit 0 - 9.

If you don't specify any arguments, TMP will display the current contents of TMP0: - TMP9:.

Options:

    /C   Clears TMP *n.*

    /S   Sets TMP *n* to *text*

    /Z   Clears all the temporary devices

4.2.199 TOAST

Purpose: Displays Windows Toast notifications

Format: TOAST /template=*n* /text1="*text*" [*options*]

    See Options below for details

Usage:

Display Windows Toast notifications, a popup window that appears on the lower right corner of the display. Unlike message boxes, Toast popups are not modal and will disappear after a few seconds. Windows will not display Toast notifications if the user has disabled notifications, either for TCC-RT or everywhere.

TOAST sets two internal command variables:

_toast
    0 - no toast active or no user response yet
    1 - user clicked on the toast
    2 - user dismissed the toast
    3 - toast timed out
    4 - application hid the toast
    5 - toast was not activated
    6 - toast failed
    7 - system does not support toasts
    8 - unhandled option
    9 - multiple texts were provided
    10 - toast notification manager initialization failure
    11 - toast could not be launched

_toast_action
    0 - user has not clicked on a button
    1 - user clicked on first button
    2 - user clicked on second button
    3 - user clicked on third button

The TOAST command exits after calling Windows to display the Toast notification. Windows will call back to TOAST with the Toast results and actions, so the _toast and _toast_action variables will not be set until the user either clicks on the Toast or it times out.

Example:

```
toast /image="console2.png" /expire=15 /action=Yes /action=No /text1="Do
you want to try to take over the world?"
```

Options:

/action="*text"*** - You can have one or more actions. Each action creates a button on the Toast window; clicking on that button will set the _toast_action internal variable.

/attribute="*text"*** - Attribution text displayed on the bottom of the Toast window.

/audio=*n*** - Windows system sound to play when the notification is displayed.
    0 - DefaultSound
    1 - IM
    2 - Mail
    3 - Reminder
    4 - SMS
    5 - Alarm
    6 - Alarm2
    7 - Alarm3
    8 - Alarm4
    9 - Alarm5
    10 - Alarm6
    11 - Alarm7
    12 - Alarm8
    13 - Alarm9
    14 - Alarm10
    15 - Call
    16 - Call1

17 - Call2
18 - Call3
19 - Call4
20 - Call5
21 - Call6
22 - Call7
23 - Call8
24 - Call9
25 - Call10

/audiostate=*n* - Specifies whether you want to display the sound (see /audio above) once, looping, or not at all.
0 - Default
1 - Silent
2 - Loop

/duration=*n* - The time to display the Toast notification
0 - Default
1 - Short
2 - Long

/expire=*n* - Number of seconds before the notification expires.

/image="*pathname***"** - The image file you want to display (for template types 0 - 3)

/template=*n* - The type of Windows Toast you want to display:
0 - An image on the left, and a string that occupies a maximum of three lines
1 - An image on the left, a bold string on the first line and a second string wrapped across the second and third lines.
2 - An image on the left, a bold string on the first and second lines, and a second string on the third line.
3 - An image on the left, a bold string on the first line, a second string on the second line, and a third string on the third line.
4 - A string that occupies a maximum of three lines
5 - A bold string on the first line and a second string wrapped across the second and third lines.
6 - A bold string on the first and second lines, and a second string on the third line.
7 - A bold string on the first line, a second string on the second line, and a third string on the third line.

/S - Create the shortcut to *TCC* required for Toast notifications. (Not valid with any other options.) This is normally done by the installer, so you shouldn't need to run TOAST /S unless the shortcut was removed.

/text1="*text***"** - Text to display in the first line (template types 0 - 7).

/text2="*text***"** - Text to display in the second line (for template types 1, 2, 3, 5, 6, and 7)

/text3="*text***"** - Text to display in the third line (for template types 3, and 7)

4.2.200 TOUCH

Purpose: Change a file's time stamps, and optionally create a file

Format: TOUCH [/A:[[-][+]rhsdaecjot] /C /CD [/D[acw][date] /E /F /I'""*text*"" /N /O:[-] acdeginorstuz /Q /R[:acw] *file* /S[[+]n] /T[acw[u]][hh:mm[:ss[.dd]]] *file*...

file	One or more files whose date and/or time stamps are to be changed.

/A:	Attribute select	/N	No action
/C	Create file	/O:	Order
/CD	Create directory	/Q	Quiet
/D	Date	/R	Reference file
/E	No error messages	/S	Subdirectories
/F	Force read-only files	/T	Time
/I	Match descriptions		

File Selection:

Supports attribute switches, extended wildcards, ranges, multiple file names, subdirectories, catalog files, and include lists.

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: option to include directories. /A:D will select directories only.

TOUCH sets three internal variables:

%_touch_dirs	The number of directories touched
%_touch_files	The number of files touched
%_touch_errors	The number of errors

Examples:

Change the last write date/time on the file *testfile.txt* to the current date/time:

```
touch testfile.txt
```

Change the creation date/time on the file *testfile.txt* to January 1, 2022 at 12:01am:

```
touch /dc2022-01-01 /tc00:01 testfile.txt
```

Options:

/A: Select only those files that have the specified attribute(s) set. See Attribute Switches for information on the attributes which can follow /A:.

You can specify **/A:=** to display a dialog to help you set individual attributes.

/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.

/CD   Create the specified directory if it does not already exist.

/D   If neither /R nor /D are specified, the current date is used. If the /D option is specified without date, TOUCH will not modify the date even if /R is also specified. If the /D option is followed by date, and /R is not specified, date is used. The date must not be quoted. If both /R and /D 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 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 for details.

/N   Display what would occur without actually doing it.

/O:...   Sort the files before processing. 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, instead of numerically when numeric substrings are included in the name or extension.
         c   Sort by compression ratio
         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 description
         o   Sort by owner
         r   Reverse the sort order for all options
         s   Sort by size
         t   Same as **d**
         u   Unsorted
         z   Same as **s**

/Q   Do not display normal messages.

/R   The /R option permits duplication of the time stamp of ***ref_file*** (which must immediately follow the /R, and can be a file or subdirectory). 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 time stamp 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.pdf %temp\tcmd.pdf
copy /u ftp://ftp.jpsoft.com/help/tcmd.pdf %temp\tcmd.pdf
```

In the above example TOUCH creates an empty file with the time stamp of your already existing help file; COPY updates the empty file if a newer version is available (beware of time stamp 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 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.

If you specify a **+** followed by a number after the /S, TOUCH will not modify any time stamps until it gets to that depth in the subdirectory tree. For example, if you have a directory tree \a\b\c\d\e, /S+2 will not modify anything in \a or \a\b.

/T   If neither /R nor /T are specified, the current time is used. If the /T option is specified without time, TOUCH will not modify the time even if /R is also specified. If the /T option is followed by time, and /R is not specified, time is used. (Time must not be quoted). If both /R and /T 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.

4.2.201 TPIPE

Purpose: Text filtering, search, substitution, conversion, and sorting

Format: See Options below.

Usage:

TPIPE does text filtering, substitution, conversion, and sorting on text files. If you don't specify an input filename, TPIPE will read from standard input if it has been redirected. If you don't specify an output filename, TPIPE will write to standard output. This is substantially slower than reading from and writing to files, but allows you to use TPIPE with pipes.

You can specify multiple filters, which will be processed in the order they appear on the command line. Do not insert any unquoted whitespace or switch characters in the arguments to an option. If you do need to pass whitespace, switch characters, or double quotes in an argument, you can quote the entire option in single back quotes.

Row and column positions start at 1.

TPIPE defaults to UTF8 encoding when loading or saving files.

If you need to process a Windows Unicode UTF-16 file, unless the filter supports Unicode directly (for example, **/simple**) you'll need to convert it to UTF-8 first (see **/unicode=...**).

Options:

- /input=filename[,subfolders[,action]]

    filename - Filename or folder to read. This can be either a disk file, file list (@filename), or CLIP:. If it is not specified, TPIPE will read from standard input.
    subfolders - How many subfolders to include (default 0):
      0 - no subfolders
      1 to 255 - subfolder(s)
      255 - all subfolders
    action - the action to take (default 1):
      1 - include the files
      2 - exclude the files
      3 - ignore the files

    You can specify multiple **/input** statements.

- /output=filename

    Filename to write. This can be either a disk file or CLIP:. If it is not specified, TPIPE will write to standard output.

- /outputfolder=directory

    Set the output filter directory.

- /inputbinary=n[,size]

    Determines how binary files are processed. The options are:

      0 - Binary files are processed (default)
      1 - Binary files are skipped
      2 - Binary files are confirmed before processing
      size - The sample size in bytes to use for identifying binary files (default 255)

□ /inputclipboardunicode=n

In clipboard mode, determines whether the input is ASCII (0) or Unicode (1). The default is 0.

□ /inputdelete=n

If 1, the input files will be deleted after processing. USE WITH CAUTION!

□ /inputprompt=n

If 1, TPIPE will prompt before processing each input files.

□ /inputpromptRO=n

If 1, TPIPE will prompt before processing read-only input files.

□ /inputstring=...

Process the string as if it were a file and return the result. This option will write the return value to STDOUT; you cannot specify an */output* argument.

□ /outputappend=n

If **n** is 1, append to the output file.

□ /outputchanged=n

Sets the output changed mode. The options are:
0 - Always output
1 - Only output modified files
2 - Delete original if modified

□ /outputmode=n

If **n**=1, TPIPE will open each output file in its associated program upon completion. If there is no association for a file, it will be opened in the default editor.

0 - Output to clipboard (all files are merged)
1 - Output to files
2 - Output to a single merged file

□ /outputopen=n

If 1, TPIPE will open each output file in its associated program upon completion.

□ /outputretaindate=n

If n is 1, retain the existing file date for the output file.

□ /clipboard

Runs the current filter with input from and output to the clipboard.

□ /filter=filename

Name of filter file to load (see /save=filename)

□ /save=filename

Saves the filter settings defined on the command line to the specified filename, and returns without executing any filters.

□ /startsubfilters

The following filters are created as sub filters, until the closing /ENDSUBFILTERS. Sub filters allow a restricted part of the entire text to be operated on by a group of filters without effecting the entire text. For example, a "Restrict to delimited fields" (CSV, Tab, Pipe, etc.) filter can pick out a range of CSV fields, and then a search/replace filter can operate JUST on the text restricted.

□ /endsubfilters

End the sub filters defined by the preceding /STARTSUBFILTERS.

□ /buffersize=n

Sets the buffer size for the preceding search/replace filter. (The default is 4096.)

□ /editdistance=n

Sets the edit distance threshhold for the preceding search/replace filter. (The default is 2.)

□ /comment=text

Add a comment to a filter file.

Text - Comment to add

⊟ /database=Mode,GenerateHeader,Timeout,Connection,InsertTable,FieldDelimiter,Qualifier

Adds a database-type filter. Database filters will change the output extension to match the format.

Mode
0 Delimited output
1 Fixed width
2 XML
3 Insert script
4 JSON output

GenerateHeader - Generates header information when True.

Timeout - SQL command timeout in seconds.

ConnectionStr - The database connection string.

InsertTable - The name of the insert table.

FieldDelimiter - The string to use between columns.

Qualifier - The string to use around string column values.

⊟ /dup=Type,MatchCase,StartColumn,Length,IncludeOne,Format

Remove or show duplicate lines. The arguments are:

Type:
0 - Remove duplicate lines
1 - Show duplicate lines

MatchCase - If 1, do case-sensitive comparisons.

StartColumn - The starting column for comparisons (the first column is 1).

Length - The Length of the comparison.

IncludeOne - If 1, include lines with a count of 1 (only for Type 1).

Format - how the output should be formatted for Type=1. Format strings are composed of plain text and format specifiers. Plain text characters are copied as-is to the resulting string.
Format specifiers have the following form:
%[index ":"][-][width][.precision]type
An optional argument index specifier
An optional left justification indicator, ["-"]
An optional width specifier, [width] (an integer). If the width of the number is less than the width specifier, it will be padded with spaces.
An optional precision specifier [precision] (an integer). If the width of the number is less than the precision, it will be left padded with 0's.
The conversion type character:
d - decimal
s - string

Percent signs in the format string should be doubled (unless you back quote the /dup=`...` argument), and the count argument must appear before the string (unless you use the index specifier). For example, "%%d %%s" shows the count followed by the string. Or, with the index specifier, "string %%1:s count %%0:d".

⊟ **/eol=input,output,length,LFString,Remove**

Add an EOL (end of line) conversion filter. The arguments are:

input:
> 0 - Unix (LF)
> 1 - Mac (CR)
> 2 - Windows (CR/LF)
> 3 - Auto
>> If you are unsure of the source, select Auto. The Auto option can detect and modify text files containing a variety of line endings.
> 4 - Fixed (use the length parameter to specify the length)
>> If you are converting a mainframe file that contains fixed length records, select "Fixed length" and enter the record length. The maximum record length is 2,147,483,647 characters. Note: If you are converting 132 column mainframe reports, you should set the fixed length to 133, because each line has a prefix character.

output:
> 0 - Unix
> 1 - Mac
> 2 - Windows
> 3 - None

length - The line length to use if input=4

LFString (optional) - The new line feed string on output when option 4 is chosen for input

Remove (optional) - Whether to remove bad EOLs (default 1)

⊟ **/file=type,MatchCase,filename**

Add a file-type filter. The arguments are:

type:
> 0 - Add left margin
> 1 - Add header
> 2 - Add footer
> 3 - Add right margin
> 4 - Remove lines that match exactly
> 5 - Retain lines that match exactly
> 6 - Remove lines matching the Perl pattern
> 7 - Retain lines matching the Perl pattern
> 8 - Add text side by side
> 9 - Add repeating text side by side
> 10 - Not Used
> 11 - Not Used
> 12 - XSLT transform

13 - Restrict to lines from list
14 - Restrict to lines NOT in list
15 - Restrict to lines matching the Perl pattern
16 - Restrict to lines NOT matching the Perl pattern
17 - Restrict to filenames patching the Perl pattern
18 - Restrict to filenames NOT matching the Perl pattern

MatchCase - If 1, do a case sensitive match (where appropriate)

filename - the filename to use

/grep=Type,IncludeLineNumbers,IncludeFilename,MatchCase,CountMatches,PatternType, UTF8,IgnoreEmpty,Pattern

Adds a Grep type line based filter. The arguments are:

Type:
0 Restrict lines matching (for subfilters)
1 Restrict lines NOT matching (for subfilters)
2 Extract matches
3 Extract matching lines (grep)
4 Extract non-matching lines (inverse grep)
5 Remove matching lines
6 Remove non-matching lines

IncludeLineNumbers - 1 to include the line number where the pattern was found

IncludeFilename - 1 to include the filename where the pattern was found

MatchCase - 1 to do a case-sensitive comparison when matching the pattern

CountMatches - 1 to output a count of the number of matches

PatternType
0 Perl pattern
1 Egrep pattern
2 Brief pattern
3 MS Word pattern

UTF8 - 1 to allow matching Unicode UTF8 characters

IgnoreEmpty - 1 to ignore empty matches

Pattern - the (regular expression) pattern to match

/head=Exclude,LinesOrBytes,Count

Add a head type filter (includes or excludes text at the beginning of the file). The arguments are:

Exclude:
0 - Include the text

1 - Exclude the text

LinesOrBytes:
0 - Measure in lines
1 - Measure in bytes

Count - the number of lines or bytes to include or exclude

⊟ /insert=type,position,string

Add an insert type filter. The arguments are:

type:
0 - Insert column
Inserts a new column of text. The position the text is inserted is determined by a column count. The leftmost column is column 1 – inserting in this column displaces all other text to the right. If the insert column given is 0, the text is inserted at the end of the line. If the insert column is negative, the text is inserted at the given position relative to the end of the line. If the insert column given is before the start of the line, or beyond the end of the line, then the text is prepended or appended to the line respectively. Note - this filter is designed for ANSI or Unicode UTF-8 data - it will not handle UTF-16 data. If you need to process UTF-16 files, convert them to UTF-8 first and then convert back to UTF-8 after doing the insertion.
1 - Insert bytes
Insert bytes at the given offset (from 0 to the size of the file).

position - the position to insert the string

string - the string to insert

⊟
/line=StartNumber,Increment,SkipBlank,DontNumberBlank,NumberFormat[,DontReset[,ResetNewFile]]

Adds a Line Number filter. The arguments are:

StartNumber - the starting line number

Increment - the amount to add for each new line number

SkipBlankIncrement - don't increase the line number for blank lines

DontNumberBlank - don't put a line number on blank lines

NumberFormat - The format to use for the line number. The format syntax is:
[-][width][.precision]d
An optional left justification indicator, ["-"]
An optional width specifier, [width] (an integer). If the width of the number is less than the width specifier, it will be padded with spaces.
An optional precision specifier [precision] (an integer). If the width of the number is less than the precision, it will be left padded with 0's.

The conversion type character:
d - decimal

DontReset - if 1, do not reset the line count at the end of the file. The default is 0.

ResetNewFile - if 1, reset the count at the start of a new file. The default is 0.

☐ /log=Filename

Log the TPIPE actions.

Filename - Name of log file

☐ /logappend=n

If n is 1, append to the log file.

☐ /maths=operation,operand

Adds a maths type filter.

operation - the operation to perform
0	+
1	-
2	*
3	div (the remainder is ignored)
4	mod (the remainder after division)
5	xor
6	and
7	or
8	not
9	shift left (0 inserted)
10	shift right (0 inserted)
11	rotate left
12	rotate right

operand - the operand to use

☐ /merge=type,filename

Adds a merge type filter (merge into single output filename). The arguments are:

type:
0 Merge into filename
1 Retain lines found in filename
2 Remove lines found in filename
3 Link filter filename

filename - the filename to use

☐ /number=type,value

Add a number-type filter. The arguments are:

type:
0 Convert Tabs to Spaces
1 Convert Spaces to Tabs
2 Word wrap (value column width)
3 Pad to width of value
4 Center in width of value
5 Right justify in width of value
6 Restrict CSV field to value
7 Restrict tab-delimited field to value
8 Truncate to width value
9 Force to width value
10 Repeat file value times
11 Restrict to blocks of length
12 Expand packed decimal (with implied decimals)
13 Expand zoned decimal (with implied decimals)
14 Expand unsigned (even-length) packed decimal
15 Expand unsigned (odd-length) packed decimal

Value - the numeric value to use

☐ /perl=BufferSize,Greedy,AllowComments,DotMatchesNewLines

Sets the Perl matching options for the immediately preceding search/replace filter.

BufferSize - The maximum buffer size to use for matches. Any match must fit into this buffer, so if you want to match larger pieces of text, increase the size of this buffer to suit. Default is 4096.

Greedy - If the pattern finds the longest match (greedy) or the shortest match. Default is false.

AllowComments - Allow comments in the Perl pattern. Default is false.

DotMatchesNewLines - Allow the '.' operator to match all characters, including new lines. Default is true.

☐
/replace=Type,MatchCase,WholeWord,CaseReplace,PromptOnReplace,Extract,FirstOnly,SkipPromptIdentical,Action,SearchStr,ReplaceStr

Adds a search and replace (find and replace) filter. Search / Replace lists discard blank search terms and terms where the replacement is identical to the search. Search / Replace lists can generate log entries (useful for debugging). Logs can optionally be output only for where replacements occurred.

The arguments are:

Type:
0 Replace
1 Pattern (old style)
2 Sounds like

3 Edit distance
4 Perl pattern
5 Brief pattern
6 Word pattern

MatchCase - Matches case when set to 1, ignores case when set to 0

WholeWord - Matches whole words only when set to 1

CaseReplace - Replaces with matching case when set to 1

PromptOnReplace - Prompts before replacing when set to 1

Extract - If 1, all non-matching text is discarded

FirstOnly - If 1, only replace the first occurrence

SkipPromptIdentical - If 1, don't bother prompting if the replacement text is identical to the original.

Action - the action to perform when found:
    0 replace
    1 remove
    2 send to subfilter
    3 send non-matching to subfilter
    4 send subpattern 1 to subfilter etc

SearchStr - the string to search for

ReplaceStr - the string to replace it with

/replacelist=Type,MatchCase,WholeWord,CaseReplace,PromptOnReplace,FirstOnly,SkipPromptIdentical,Simultaneous,LongestFirst,Filename

Add a search and replace list, using search and replace pairs from the specified file.

Type:
    0 Replace
    1 Pattern (old style)
    2 Sounds like
    3 Edit distance
    4 Perl pattern
    5 Brief pattern
    6 Word pattern

MatchCase - Matches case when set to 1, ignores case when set to 0

WholeWord - Matches whole words only when set to 1

CaseReplace - Replaces with matching case when set to 1

PromptOnReplace - Prompts before replacing when set to 1

FirstOnly - If 1, only replace the first occurrence

SkipPromptIdentical - If 1, don't bother prompting if the replacement text is identical to the original.

Simultaneous - If 1, all search strings are scanned for simultaneously instead of consecutively. (This is useful if the search strings and results strings overlap.)

LongestFirst - If 1, searches for long phrases (most specific) before short phrases (least specific) - this is generally used for translations.

Filename - The file to load search/replace pairs from. If the file extension is .XLS or .XLSX, the file is assumed to be Excel format, if the extension is .TAB the file is assumed to have tab-delimited values, and any other extension (including .CSV) is assumed to have Comma-Separated Values. The filename can contain environment variables enclosed in % signs e.g. %TEMP%\myfile.txt. TPIPE corrects any doubled backslashes.

☐ /run=InputFileName,OutputFileName,"CommandLine"

Adds a Run External Program filter. The arguments are:

InputFilename - the filename that TextPipe should read from after the External Program writes to it.
OutputFilename - the filename that TextPipe should write to for the External Program to read in.
CommandLine - the command line of the program to run. Should include double quotes around the entire command line.

☐ /script=language,timeout,code

Adds an ActiveX script filter.

language: The language of the script

timeout: The command timeout in seconds

script: The code

☐
/selection=Type,Locate,Param1,Param2,MoveTo,Delimiter,CustomDelimiter,HasHeader[,ProcessIndividually[,ExcludeDelimiter[,ExcludeQuotes]]]

Type - The type of filter to add:
0 – Remove column:
This filter is used to remove columns of text, given a column specification that describes the position of the column relative to the start or end of the line, and the width of the column. There are several ways to specify the columns (Locate,Param1,Param2) to remove:
0 - Start column, End column. This removes all text including and between the specified columns. Useful for removing column in fixed width data files.
1 - Start column, width. Removes Width characters starting from (and including) column Start.

2 - End column, width. Removes Width characters backwards starting from (and including) column End.

3 - Start column to end of line. Removes all characters from the Start column to the very end of the line. Useful for making a file a uniform width.

4 - Width to end of line. Removes Width characters backwards starting from (and including) the last column.

Note - if you are removing more than one column range, it is easiest to remove ranges from right-to-left so that the position of the columns doesn't change.

1 - Restrict lines (restriction filters require sub filters to have any effect)

2 - Restrict columns (restriction filters require sub filters to have any effect)

3 - Restrict to bytes (restriction filters require sub filters to have any effect)

4 - Restrict to delimited fields (CSV, Tab, Pipe, etc.)

6 – Remove lines:

This filter removes a range of lines. There are several ways to specify the lines (Locate,Param1,Param2) to remove:

0 - Start line, End line. This removes all lines including and between the specified lines.

1 - Start line, width. Removes Width lines starting from (and including) line Start.

2 - End line, width. Removes Width lines backwards starting from (and including) line End.

3 - Start line to end of line. Removes all lines from the Start line to the very end of the line.

4 - Width to end of line. Removes Width lines backwards starting from (and including) the last line.

7 – Remove delimited fields (CSV, Tab, Pipe, etc.):

This filter is used to remove fields delimited by a given character. You can choose a predefined delimiter character (Delimiter), or select your own (CustomDelimiter). The trailing delimiter (if any) is also removed. When Comma (.csv) is chosen, TPIPE automatically handles single and double quoted strings, with embedded line feeds.

First Row Contains Field Names

If the first line of the file contains Field Names, set HasHeader to 1 so that TPIPE can count how many fields are expected. It can then determine if a field has embedded CR/LF characters and spans multiple lines. TPIPE can also determine this without a header if the fields are properly double-quoted - TPIPE will notice the missing double quote and continue reading the record from the following line.

Remove Fields

There are several ways to specify the fields (Locate,Param1,Param2) to remove:

0 - Start field, end field. This removes all text including and between the specified fields.

1 - Start field, width. Removes Width fields starting from (and including) field Start.

2 - End field, width. Removes Width fields backwards starting from (and including) field End.

3 - Start field to end of line. Removes all fields from the Start field to the very end of the line.

4 - Width to end of line. Removes Width fields backwards starting from (and including) the last field.

Note - if you are removing more than one field range, it is easiest to remove ranges from right-to-left so that the position of the fields doesn't change.

9 – Move columns:

TPIPE will move columns to a new position on the line. The new position (MoveTo) is specified assuming that the moved columns have been removed from the line.

10 – Move delimited fields (CSV, Tab, Pipe, etc.):

TPIPE will move CSV-delimited fields to a new position on the line. The new position (MoveTo) is specified assuming that the moved fields have been removed from the line. TPIPE ensures that all the delimiters on the line are correctly maintained, both at the end of the line and where the moved fields are inserted. Note - this filter is designed for ANSI or Unicode UTF-8 data - it will not handle UTF-16 data. You will need to convert UTF-16 files to UTF-8 first, do the selection, and then convert back to UTF-16.

12 – Copy columns:

TPIPE will copy columns to a new position (MoveTo) on the line. Note - this filter is designed for ANSI or Unicode UTF-8 data - it will not handle UTF-16 data. You will need to convert UTF-16 files to UTF-8 first, do the selection, and then convert back to UTF-16.

13 – Copy delimited fields (CSV, Tab, Pipe, etc.):

TPIPE will copy CSV-delimited fields to a new position (MoveTo) on the line. TPIPE ensures that all the delimiters on the line are correctly maintained, both at the end of the line and where the copied fields are inserted. Note - this filter is designed for ANSI or Unicode UTF-8 data - it will not handle UTF-16 data. You will need to convert UTF-16 files to UTF-8 first, do the selection, and then convert back to UTF-16.

17 – Remove byte range:

This filter is used to remove a range of bytes. There are several different ways to specify the bytes (Locate,Param1,Param2) to remove:

0 - Start byte, end byte. This removes all text including and between the specified byte.

1 - Start byte, width. Removes Width byte starting from (and including) the start byte.

2 - End byte, width. Removes Width fields backwards starting from (and including) byte End.

3 - Start byte to end of file. Removes all fields from the Start byte to the very end of the file.

4 - Width to end of file. Removes Width fields backwards starting from (and including) the last byte.

Note - if you are removing more than one byte range, it is easiest to remove ranges from right-to-left so that the position of the bytes doesn't change.

Locate - How to determine which areas to affect:

0 - Restrict %d .. %d
1 - Restrict %1:d starting at %0:d
2 - Restrict %1:d starting at END - %0:d
3 - Restrict %d .. END - %d
4 - Restrict END - %d .. END - %d

Param1, **Param2** - The integer values for the Locate method.

MoveTo - The integer value where to move or copy the columns or fields to (first columns or field is 1).

Delimiter - The index of the standard delimiter to use:
    0 - Comma
    1 - Tab
    2 - Semicolon
    3 - Pipe (|)
    4 - Space
    5 - Custom

CustomDelimiter - The custom delimiter to use (if Delimiter == 5). This should be a quoted string; if you are not using a custom delimiter then set this field to **""**.

HasHeader - 1 if the file's first row is a header row, 0 if not.

ProcessIndividually - Whether to apply sub filters to each CSV or Tab field individually (1), or to the fields as one string value (0). The default is 0.

ExcludeDelimiter - Whether or not to include the comma or Tab field delimiter when passing the field to the sub filter. Defaults to 0.

ExcludeQuotes - Whether or not to include the CSV quotes that may surround the field when passing the field to the subfilter. Defaults to 1.

⊟ /selection2=type, columnSpec,moveTo,processIndividually,excludeDelimiter,excludeQuotes,delimiter[,customDelimiter,hasHeader]

Type - the type of filter to add

    0 - Delete column
    1 - Restrict lines
    2 - Restrict columns
    3 - Restrict to bytes
    4 - Restrict to delimited fields (CSV, Tab, Pipe etc)
    5 - unused
    6 - Remove lines
    7 - Remove delimited fields (CSV, Tab, Pipe etc)
    9 - Move columns
    10 - Move delimited fields (CSV, Tab, Pipe etc)
    12 - Copy columns
    13 - Copy delimited fields (CSV, Tab, Pipe etc)
    17 - Remove Byte Range
    18 - Extract fields

columnSpec - the double-quoted list of items to remove e.g. "1..10, 16, 20"

moveTo - (integer) where to move or copy the columns or fields to.

processIndividually - whether or not to apply sub filters to each CSV or Tab field individually, or to the fields as one string value.

excludeDelimiter - whether or not to include the comma or Tab field delimiter when passing the field to the sub filter.

excludeQuotes - whether or not to include the CSV quotes that may surround the field when passing the field to the sub filter.

delimiter - (optional) the index of the standard delimiter to use, default 0 for CSV

> 0 - Comma
> 1 - Tab
> 2 - Semicolon
> 3 - Pipe (|)
> 4 - Space
> 5 - Custom

customDelimiter - (optional) the double quoted custom delimiter to use; the default is blank.

hasHeader - (optional) 1 if the file's first row is a header row, default 0.

☐ /simple=n[u]

Adds a simple filter type. n is the type of filter to add, and for those filters that support it, u indicates that the filter will be dealing with Unicode data.

> 1 – Convert ASCII to EBCDIC
> EBCDIC is the character collating sequence commonly used on mainframes. Some characters cannot be converted because they exist in one character set but not the other.
> 2 – Convert EBCDIC to ASCII
> 3 – Convert ANSI to OEM
> Converts from ANSI to ASCII/OEM. ANSI is an 8-bit character set used by Windows, and it includes all accentuated Roman characters used by non-English languages like French, German and Spanish. (Windows uses UTF-16LE for all of its internal APIs, and converts to ANSI if the user is using raster fonts or ANSI files.) ASCII/OEM is an extension of the original IBM character set where various non-essential characters are replaced by language-specific accentuated characters. Different ASCII/OEM character sets are not compatible. They must be converted to ANSI and then back to the correct ASCII/OEM character set to be readable.
> 4 – Convert OEM to ANSI
> 5 – Convert to UPPERCASE
> Forces all text to UPPERCASE. To make the conversion, the function uses the current language selected by the user in the system Control Panel. If no language has been selected, TPIPE uses the Windows internal default mapping.
> 6 – Convert to lowercase
> Forces all text to lowercase. To make the conversion, the function uses the current language selected by the user in the system Control Panel. If no language has been selected, TPIPE uses the Windows internal default mapping.
> 7 – Convert to Title Case
> Converts all text to Title Case -- i.e., the first letter of every word is capitalized, and all other letters are forced to lower case. This routine calculates a table of upper and lower case letters on TPIPE startup, and this determination is based on the semantics of the language selected in Control Panel.

8 – Convert to Sentence Case
> Converts all text to Sentence case ie the first word in every sentence is capitalized, all other letters are left as is. Sentences start after periods, exclamation marks, colons, question marks, quotes, parentheses and angle brackets (.!:?'"<().

9 – Convert to tOGGLE cASE
> tOGGLES tHE cASE of all text -- i.ee, all UPPERCASE characters are converted to lowercase and vice-versa.

10 – Remove blank lines
> Removes blank lines. Note, lines with spaces or tabs are not removed. Use the Remove Blanks From Start Of Line filters first to rectify this.

11 – Remove blanks from End of Line
> Removes spaces and tabs from the end of every line.

12 – Remove blanks from Start of Line
> Removes spaces and tabs from the start of every line.

13 – Remove binary characters
> Removes binary characters such as those higher than ASCII code 127, and those less than ASCII code 32 except for carriage returns (ASCII code 13) and line feeds (ASCII code 10).
> This filter if very useful if you have a corrupted text file, or if you just want to see what text is inside a binary file. The binary information is removed, leaving you with just the text.

14 – Remove ANSI codes
> ANSI (American National Standards Institute) codes are included in various streams of information, to provide a remote computer with control over cursor positioning, text attributes, etc. They are also used in connections between minicomputers and mainframe computers and the terminals connected to them.
>
> The need to use an ANSI filter can be recognized when something like the following example shows up in a file viewed in a text editor:
>
> <[0;1;4mas<[m - MC88000 assembler
>
> In this example, the "as" near the beginning is displayed in a different color than the rest of the line when the ANSI codes are properly processed. The Escape (ASCII 27) codes above have been replaced by the < symbol to make this line printable.
>
> The Remove ANSI Escape Sequences filter can be used to filter out these codes and "clean up" the text so that it can be used in standard fashions such as copying and pasting into a word processor. On Unix machines the man (manual) help utility will only allow page-by-page browsing through a file in a forward direction. By piping the man output to a text file, transferring it to a DOS machine, and running it through the Remove ANSI Escape Sequences filter (and the Convert EOL filter - Unix to DOS if desired), a standard DOS editor can be used for browsing through the file, quoting from it, etc.

15 – Convert IBM drawing characters
> IBM drawing characters in the upper ASCII range (128-255) are commonly used to draw lines and boxes, single and double line borders, shaded characters etc. Many devices (such as printers, non-IBM computers etc.) do not support the display of these characters. This filter converts them to standard ASCII characters (+, - and |) that all computers can display.

16 – Remove HTML and SGML

Use this filter to convert HTML documents to a readable format. This filter removes HTML and XML markup tags i.e. everything including and between <> brackets.

17 – Remove backspaces

Remove backspaces, i.e. all ASCII code 8's.

18 – Resolve backspaces

Resolve backspaces -- i.e., remove both the backspaces and the characters prior to the backspaces that would have been deleted.

19 – Remove multiple whitespace

Removes sequences of multiple spaces or tabs and replaces them with a single space.

20 – UUEncode

Usually used for transmitting binary files inside an email. Files of this type are usually given an extension of *.uue*. Warning – UUencoded text may be corrupted when passing over a mainframe mail gateway. To avoid corruption, use Mime Base 64 or XXEncode.

21 – Hex Encode

A very simple encoding of a file. Usually used for small files, because it uses a large amount of space. The benefit is that the file is very easy to encode/decode, and the file cannot be corrupted passing through mail gateways.

22 – Hex Decode

Converts a file from its hex representation back to binary. The file to be decoded MUST NOT have any extra characters at the start or end if it is to be successfully processed.

23 – MIME Encode (Base 64)

Used for binary data. Files of this type are usually given an extension of *.b64*.

24 – MIME Decode (Base 64)

Used for binary data. Files of this type are usually given an extension of *.b64*. The file to be decoded MUST NOT have any extra characters at the start or end if it is to be successfully processed.

25 – MIME Encode (Quoted printable)

Quoted printable is used for text that is mainly readable, but may contain special characters with accents etc.

26 – MIME Decode (Quoted printable)

The inverse of the above encoding.

27 – UUDecode

Mail attachments can be uuencoded, use this filter to convert the file back to its correct form. Files of this type are usually given an extension of .uue.

28 – Extract email addresses

Extract email addresses. This filter searches for email addresses of the form user@server.domain, and writes them out one per line (using a DOS line feed, CR/LF). Usually this filter is followed by a filter to remove duplicate lines, and then by a Search and Replace filter, searching for \013\010 and replacing with a comma or semi-colon, depending on the email address separator used by your email software.

29 – Unscramble (ROT13)

This is a simple email encoding usually used to disguise text that some people may find offensive. The encoding is totally reversible (applying it twice removes the encoding). Only alpha characters are affected (A..Z and a..z).

30 – Hex dump

This changes the text to lines consisting of 16 bytes each. Each line has an 8 hex digit file index, 16 bytes (in hex) and the ASCII representation:

```
00000000 65 67 69 6E 0D 0A 20 20 20 20 20 20 61 64 64 72
egin........addr
00000010 65 73 73 20 3A 3D 20 0D 0A 20 20 20 20 20 20 20
ess.:=.........
00000020 20 64 65 63 54 6F 48 65 78 53 74 72 28 20 28 66
.decToHexStr(.(f
```

This filter is very useful for identifying special characters to search and replace.

32 – XXEncode

Essentially identical to UUEncode except that the character set used is different to allow it to pass through EBCDIC gateways without corruption. The XXencoding implemented by TPIPE uses the following characters:

```
+-
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijkl
mnopqrstuvwxyz
```

33 – XXDecode

Essentially identical to UUDecode except that the character set used is different to allow it to pass through EBCDIC gateways without corruption.

34 – Reverse line order

The order of the input lines is reversed i.e. the last line comes out first and the first line comes out last. A file is read entirely into RAM before being reversed, so be wary of reversing files that are larger than your machine's RAM size.

35 – Remove email headers

This filter removes the email headers that accompany emails exported to a text format. The email headers are the lines such as To:, From:, Subject: and various other message headers added by all the servers through which your email passes before it gets to its destination.

36 – Decimal dump

This changes the text to lines consisting of 10 bytes each. Each line has a 10 decimal digit file index, 10 bytes (in decimal) and the ASCII representation:

```
0000000000 080 108 101 097 115 101 032 102 101 101
Please fee
0000000010 108 032 102 114 101 101 032 116 111 032 l free
to
0000000020 099 111 109 109 101 110 116 013 010 111
comment..o
```

This filter is very useful for identifying special characters to search and replace.

37 – HTTP Encode

This filter is used to encode text for use in an HTTP header – a (usually) small piece of text that accompanies a web page request to a web server. This filter is very useful for debugging CGI scripts because it can create HTTP requests in the correct form. HTTP encoded text usually looks like the following:

```
a+%28usually%
29+small+piece+of+text+that+accompanies+a+web+page+requ
est+to+a+web+server.+This+filter+is+very+
```

38 – HTTP Decode

This filter is used to decode text from an HTTP header – a (usually) small piece of text that accompanies a web page request to a web server.

39 – Randomize lines

This filter put lines into random order. This is useful when a random sample of data is required for statistical purposes - just follow this filter with a head/tail of file filter (/head or /tail). The lines output will differ from one run to the next; the order is determined by a pseudo-random number generator.

40 – Create word list

This filter takes all the incoming words and outputs them one per line. This can be used to generate word lists for Indexes, encryption programs etc. Hyphenated words are recognized as single words, provided that they aren't broken across lines. To get around this limitation, use a Search and Replace filter to replace hyphens followed by line feeds with just a hyphen. Normally you would follow this filter with a remove duplicates filter, or alternatively, a Count Duplicate Lines filter (with Include counts of 1).

> catch22 – a word
> 24-7 – a word
> twenty-four – a word
> 5th – a word
> ice cream – two words

Commas or periods after words are treated as word separators.

41 – Reverse each line

Each line is output reversed from left to right. This can be useful to extract domain names from web site log files - use this filter to reverse each line, use an extract matches filter of [\w\d]+\.[\w\d]+ to extract each domain name, then reverse each line again. Note: This filter will NOT work on Unicode or UTF-8 data. It will only work on single-byte data such as ASCII or ANSI.

42 – Convert to RanDOm case

This filter randomly changes the case of characters. This routine calculates a table of upper and lower case letters on TPIPE startup, and this determination is based on the semantics of the language selected in the Windows Control Panel.

Running this filter again will generate different results; for example:

> 1. ranDoMIze cASe
> 2. RanDOmIZE case
> 3. randOMIZE casE

43 – Extract URLs

Extract URLs. This filter lists mailto:, http://, https://, ftp://, ftps://, nntp:, skype:, call:, and gopher:// URLs one per line.

44 – ANSI to Unicode

Converts single byte ANSI characters to double byte Unicode characters. This filter can be useful if you want to send a text file to someone using a language other than your own. This filter is often followed by an Add Header filter, to add a Unicode byte order mark (BOM), \xFF\xFE.

45 – Unicode to ANSI

Converts double byte Unicode characters to single byte ANSI characters. This filter can be useful if you want to send a text file to someone using a language other than your own. This filter is often followed by a Remove start or end of file filter, to either remove the first two bytes of Unicode (before the conversion) or the first byte of ANSI

(after the conversion), to remove the leading Unicode byte order mark (BOM).

46 – Display debug window
A debug filter is very handy for debugging filters. When text is passed through this filter, it places the output into a window so that you can see what the text looks like at that stage of the filtering process.

47 – Word concordance
This filter generates a word concordance. A word concordance shows the context or surrounding words for a given set of words in a dictionary.

48 – Remove all
This filter removes all text. Unlike a pattern match filter that matches everything and then throws it away, this filter is far more efficient, especially for large files, as it signals completion back to the input filter so only the first chunk of a multi-gigabyte file will ever get processed. It is useful in two main situations

> 1. Inside a subfilter, it prevents any of the subfiltered text from re-entering the text stream. So you could restrict to lines matching a pattern, output the matching lines to a new file, and then remove them.
> 2. To remove all of the text of a file, then use an Add Header filter with the @fullInputFilename macro to obtain the name of the file.

Note: An Add Left Margin or Add Right Margin filter will not work after a Remove All filter, as they require an actual line to trigger them. Instead, use an Add Header or Add Footer filter.

49 – Restrict to each line in turn
This filter restricts sub filters to operate on each line in turn. This filter is used for its side effect of limiting the matched text to a single line at most.

50 – Convert CSV to Tab-delimited
Converts CSV data (quoted or unquoted) to tab-delimited form. It's preferable to use a file with column headers, because then TPIPE can easily determine if the fields have embedded CR/LFs in them. If the data is properly quoted then TPIPE will determine this automatically. TPIPE will eliminate unnecessary quotes.

51 – Convert CSV to XML
Converts CSV data (quoted or unquoted) to XML form. It's preferable to use a file with column headers, because then TPIPE can easily determine if the fields have embedded CR/LFs in them. If the data is properly quoted then TPIPE will determine this automatically. TPIPE correctly escapes **< > " '** and **&** in the data to the corresponding XML entity. If your data contains invalid XML characters such as ASCII 26 (End-of-file, hex \x1A), follow this filter with a search/replace filter to remove \x1A and replace with nothing.

52 – Convert Tab-delimited to CSV
Converts Tab-delimited data to CSV data. It's preferable to use a file with column headers, because then TPIPE can easily determine if the fields have embedded CR/LFs in them. TPIPE cannot determine this without column headers.

53 – Convert Tab-delimited to XML
Converts Tab-delimited data to XML data. It's preferable to use a file with column headers (/simple=55), because then TPIPE can easily determine if the fields have embedded CR/LFs in them. TPIPE cannot determine this without column headers. TPIPE correctly escapes **< > " '** and **&** in the data to the corresponding XML entity. If your data

contains invalid XML characters such as ASCII 26 (End-of-file, hex \x1A), follow this filter with a search/replace filter to remove \x1A and replace with nothing.

54 – Convert CSV (with column headers) to XML
See description for 51 – Convert CSV to XML.

55 – Convert Tab-delimited (with column headers) to XML
See description for 53 – Convert Tab-delimited to XML.

56 – Convert CSV (with column headers) to Tab-delimited
See description for 50 – Convert CSV to Tab-delimited.

57 – Convert Tab-delimited (with column headers) to CSV
See description for 52 – Convert Tab-delimited to CSV.

58 – Restrict to file name
This filter applies its subfilters only to files with filenames (ie drive + path + filename) matching or not matching a pattern or list of patterns. This is very handy for only applying a Convert Word Documents to Text filter only to files matching the pattern
\.DOC$
With the appropriate pattern, this filter can also be used to control subfilters based on filename, folder and drive. Note that this filter uses case-insensitive Perl regular expressions, not Windows wildcards.

59 – Convert Word documents to (UTF8) text
This filter takes ALL incoming documents, opens them with Microsoft Word, and outputs them as text files. This can be used to process a set of Word Documents to text file format. After this filter you can add search and replace filters or any other filters you choose.
This filter requires Microsoft Word 98 or higher to be installed. If you wish to convert documents other than the default .DOC files, you may also need to install Word's conversion filters. If Word cannot be started automatically TPIPE will prompt you to start it manually before continuing.
Unless you know that all documents being processed are Word documents (e.g. by using a wildcard of *.doc in the Files to Process tab), you should restrict this filter to only files matching the pattern:
\.DOC$

60 – Swap UTF-16 word order
This filter swaps pairs of bytes
e.g.

Byte number	1	2	3	4	5	6	7	8
Input File	FF	FE	00	20	00	31	00	3
Output File	FE	FF	20	00	31	00	32	0

This is commonly used to transform big-endian or little-endian Unicode files so that other programs can use them.

61 – Swap UTF-32 word order
This filter swaps groups of 2-byte words.
e.g.

Byte number	1	2	3	4	5	6	7	8
Input File	FF	FE	00	00	00	31	00	0
Output File	00	00	FE	FF	00	00	31	0

This is commonly used to transform big-endian or little-endian Unicode files so that other programs can use them.

62 – Remove BOM (Byte Order Mark)
This filter removes the Unicode Byte Order Mark from the start of Unicode files, if it is present.

Bytes removed	Description
00 00 FE FF	UTF-32, big-endian
FF FE 00 00	UTF-32, little-endian
FE FF	UTF-16, big-endian
FF FE	UTF-16, little-endian
EF BB BF	UTF-8

63 – Make Big Endian

Converts a Little Endian Unicode file into a Big Endian Unicode file

e.g.

Input file	Output file
00 00 FE FF 00 00 00 4D	Unchanged
FE FF 4E 8C	Unchanged
FF FE 00 00 4D 00 00 00	00 00 FE FF 00 00 00 4D
FF FE 8C 4E	FE FF 4E 8C

Note - the file MUST start with a Byte Order Mark (BOM) for it to be correctly identified.

64 – Make Little Endian

Converts a Big Endian Unicode file into a Little Endian Unicode file

e.g.

Input file	Output file
00 00 FE FF 00 00 00 4D	FF FF 00 00 4D 00 00 00
FE FF 4E 8C	FF FE 8C 4E
FF FE 00 00 4D 00 00 00	Unchanged
FF FE 8C 4E	Unchanged

Note - the file MUST start with a Byte Order Mark (BOM) for it to be correctly identified.

65 – Compress to Packed Decimal

This filter compresses EBCDIC numeric data (optional leading sign, numbers and periods) to an EBCDIC packed decimal field (also known as Comp-3).

There are several notes to keep in mind when using this filter:

1. You MUST use this filter inside a Restrict to Byte Range filter. The field WIDTH is then set by the containing filter.

2. Compressing a field will decrease your output record length, so ensure you allow for this. A good strategy to avoid problems is to first compress the rightmost field, then work your work back to the leftmost field. This prevents the field column positions from changing and makes the file easier to work with.

This filter will add hex 'B' to negative fields, hex 'C' to positive fields and hex 'F' to unsigned fields. If these codes don't match what your target needs, use a column or CSV restriction to apply a search/replace.

66 – Compress to Zoned Decimal

This filter expands an EBCDIC zoned decimal field to a raw EBCDIC number with a sign. Typically this filter is then followed by a Convert EBCDIC to ASCII filter - after all other fields have been expanded as well.

There are several notes to keep in mind when using this filter:

1. You MUST use this filter inside a Restrict to Byte Range filter. The field WIDTH is then set by the containing filter.

2. Expanding a field will increase your output record length, so ensure you allow for this. A good strategy to avoid problems is to first expand the rightmost field, then work your work back to the leftmost field. This prevents the field column positions from changing and makes the file easier to work with.

67 – Expand Binary Number to EBCDIC
This filter expands a series of digits stored in binary (BIG ENDIAN) form. The maximum width is 8 bytes.
There are several notes to keep in mind when using this filter:
1. You MUST use this filter inside a Restrict to Byte Range filter. The field WIDTH is then set by the containing filter.
2. Expanding a field will increase your output record length, so ensure you allow for this. A good strategy to avoid problems is to first expand the rightmost field, then work your work back to the leftmost field. This prevents the field column positions from changing and makes the file easier to work with.
3. If the data is stored in LITTLE ENDIAN order, use a Reverse filter inside the Restriction prior to the Expand Binary Numbers filter.

68 – Expand Binary Number to ASCII
This filter expands a series of digits stored in binary (BIG ENDIAN) form. The maximum width is 8 bytes.
There are several notes to keep in mind when using this filter:
1. You MUST use this filter inside a Restrict to Byte Range filter. The field WIDTH is then set by the containing filter.
2. Expanding a field will increase your output record length, so ensure you allow for this. A good strategy to avoid problems is to first expand the rightmost field, then work your work back to the leftmost field. This prevents the field column positions from changing and makes the file easier to work with.
3. If the data is stored in LITTLE ENDIAN order, use a Reverse filter inside the Restriction prior to the Expand Binary Numbers filter.

69 – NFC - Canonical Decomposition, followed by Canonical Composition
Applies a Unicode NFC - Canonical Decomposition, followed by Canonical Composition transformation to incoming Unicode text (UTF16-LE). Output is also Unicode UTF16-LE.

70 – NFD - Canonical Decomposition
Applies a Unicode NFD - Canonical Decomposition transformation to incoming Unicode text (UTF16-LE). Output is also Unicode UTF16-LE.

71 – NFKD - Compatibility Decomposition
Applies a Unicode NFKD - Compatibility Decomposition transformation to incoming Unicode text (UTF16-LE). Output is also Unicode UTF16-LE.

72 – NFKC - Compatibility Decomposition, followed by Canonical Composition
Applies a Unicode NFKC - Compatibility Decomposition, followed by Canonical Composition transformation to incoming Unicode text (UTF16-LE). Output is also Unicode UTF16-LE.

73 – Decompose

74 – Compose
Applies a Unicode Compose transformation to incoming Unicode text (UTF16-LE). The output is also Unicode UTF16-LE.

75 – Convert numeric HTML Entities to text

This filter converts decimal/hex numeric HTML/XML entities to plain text. For example:

® → ®

® → ®

Typically, the input file is ANSI (single byte) format. This filter will output UTF-8 characters for high-value entities e.g. ᠀ The best approach is to first convert the file from ANSI to UTF-8 (/unicode), then apply this filter.

76 – Convert PDF documents to (UTF8) text

This filter takes ALL incoming documents and converts them from PDF to text. Most of the formatting will be lost.

77 – Restrict to ANSI files

78 – Restrict to Unicode UTF16 files

79 – Restrict to Unicode UTF32 files

80 – Convert Excel spreadsheets to (UTF8) text

This filter takes ALL incoming documents, opens them with Microsoft Excel, and outputs them as CSV (comma-delimited) files (hidden worksheets will be ignored). After running this filter, you can add search and replace filters or any other filters you choose, such as convert the data to Tab-delimited or XML.

This filter requires Microsoft Excel 98 or higher to be installed. If you wish to convert documents other than the default .XLS files, you may also need to install Excel's conversion filters.

Unless you know that all documents being processed are Excel documents (e.g. by using a wildcard of *.xls in the Files to Process tab), you should restrict (/simple=58) this filter to only files matching the pattern

\.XLS$

81 - Shred file

82 - Unicode to escaped ASCII

83 - Restrict to Unicode files

84 - T-filter

The T-Filter allows you to process the same output in multiple ways. You can create a subfilter, and add filters to create the desired output. When this side of the T has finished processing, the data is discarded and the original text continues processing as though the T-filter did not exist.

85 - Convert decimal/hex numeric HTML/XML entities and entity names to text (i.e., &#174 -> ®, or ® -> ®). This filter outputs UTF-8 characters for high-value entities.

86 - Convert JSON to Tab

87 - Convert Tab to JSON

88 - Convert Word documents to RTF

/Simple has some redaction filters which are designed to work inside restriction filters.

89 - Remove diacritics

91 - Redact x-over text

92 - Redact x-over digits

93 - Redact x-over all but last 4 digits

94 - Redact x-over non-blanks

95 - Replace with blanks

96 - Redact with pseudo NHS

97 - Redact with pseudo SSN

98 - Redact with pseudo bank number

/sort=Type,Reverse,RemoveDuplicates,StartColumn,Length

Sort text files.

Type - the sort type
0 - ANSI sort
1 - ANSI sort (case sensitive)
2 - ASCII sort
3 - ASCII sort (case sensitive)
4 - Numeric sort
5 - Sort by length of line
6 - sort by date and time
7 - sort by date
8 - sort by time
9 - UTF8 sort (case insensitive)
10 - UTF8 sort (case sensitive)

Reverse - If 1, sort in descending order; if 0, sort in ascending order

RemoveDuplicates - If 1, remove duplicate lines; if 0 keep duplicate lines

StartColumn - The column in the line to begin the comparisons

Length - The length of the comparison

/split=type,SplitSize,SplitChar,SplitCharPos,SplitCharCount,SplitLines,SplitFilename[,FirstFileNumber,PreventOverload]

Adds a split type filter. The arguments are:

type:
0 Split at a given size
1 Split at a given character
2 Split at a given number of lines

splitSize - the size file to split at

splitChar - the character to split at

splitCharPos
0 Split before the character (it goes into the next file)
1 Split after the character (it remains in the first file)
2 Split on top of the character (remove it)

SplitCharCount - the number of times to see SplitChar before splitting

SplitLines - (optional) split after a given number of lines, default 60

SplitFilename - (optional) the name to give to each output split file. /split will append a "%3.3d" format specifier to the name; i.e. SplitFilename of "foo.txt" will generate output files named "foo.txt.000", "foo.txt.001", etc. If

you don't specify a SplitFilename, /split will use the input filename as the base.

FirstFileNumber - (optional) the number of the first file; default is 0

PreventOverload - (optional) true to prevent more than 10,000 files in one folder, default false

The split file filter will remove the last file if it is empty.

- /string=type,MatchCase,string

Add a string-type filter. The arguments are:

type:
    0 - Add left margin
    1 - Add header
    2 - Add footer
    3 - Add right margin
    4 - Remove lines that match exactly
    5 - Retain lines that match exactly
    6 - Remove lines matching the Perl pattern
    7 - Retain lines matching the Perl pattern
    8 - Add text side by side
    9 - Add repeating text side by side
    10 - Not Used
    11 - Not Used
    12 - XSLT transform
    13 - Restrict to lines from list
    14 - Restrict to lines NOT in list
    15 - Restrict to lines matching the Perl pattern
    16 - Restrict to lines NOT matching the Perl pattern
    17 - Restrict to filenames patching the Perl pattern
    18 - Restrict to filenames NOT matching the Perl pattern

matchCase - case sensitive or not (where appropriate)

string - the string to use

- /tail=Exclude,LinesOrBytes,Count

Add a tail type filter (includes or excludes text at the end of the file). The arguments are:

Exclude:
    0 - Include the text
    1 - Exclude the text

LinesOrBytes:
    0 - Measure in lines
    1 - Measure in bytes

Count - the number of lines or bytes to include or exclude

- /unicode=input,output

Convert the file to or from Unicode. **input** is the encoding for the input file; **output** is the encoding for the output file. The possible values are:

> UTF-16LE
> UTF-16BE
> UTF-32LE
> UTF-32BE
> UTF-8
> ANSI
> ASCII
> CP*nnn*, where *nnn* is a Windows code page (for example, CP437 or CP1251).

TPIPE handles files internally as UTF-8, so if you want to process a Windows UTF-16LE file, you'll need to convert it to UTF-8 first, then apply the desired filters, and convert it back to UTF-16LE. For example, to wrap a Unicode file at column 80:

> tpipe /input=inputname /output=outputname /unicode=UTF-16LE,UTF-8 /number=2,80 /unicode=UTF-8,UTF-16LE

⊟ /xml=Type,IncludeText,IncludeQuotes,MatchCase,BufferSize,Tag,Attribute,EndTag

Adds an HTML / XML filter. The arguments are:

Type - the operation to perform:
  0 restrict to an element
  1 restrict to an attribute
  2 restrict to between tags

IncludeText - whether to include the find string in the restriction result (default false)

IncludeQuotes - whether to include surrounding quotes in the attribute result or not (default false)

MatchCase - match case exactly or not (default false)

BufferSize - the maximum expected size of the match (default 32768)

Tag - the element or start tag to find

Attribute - the attribute to find

EndTag - the endTag to find

4.2.202 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).

4.2.203 TREE

Purpose: Display a graphical directory tree

Format: TREE [/\ /A:[-|+]rhsadecijopt /A /B /D /F /H /L /Nj /O:[-]acdeginorstuz /P[n] /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)	/L (colorize display)
/A(SCII)	/O(rder)
/B(are)	/P(ause)
/D(escriptions)	/S (file size)
/F(iles)	/Sn (subdirectory depth)
/H(idden directories)	/T(ime and date)
/N (disable option)	/Z (file size)

File Selection:

Supports attribute switches, extended wildcards, ranges (with **/F**), and multiple file names.

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, or configure TCC-RT 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 by using standard redirection symbols. Be sure to review the /A option before attempting to print the TREE output. The options discussed below specify the amount of information included in the display.

Colors

TREE can display each file name and the associated file information in a different color, depending on the file's extension, attributes, or matching range.

To choose the display colors, you must either use the SET command to create an environment variable called COLORDIR, or use the Directory Colors 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 option. You may find it useful to use the COLORDIR variable for experimenting, then to set permanent directory colors with the Directory Colors option.

The format for both the COLORDIR environment variable and the Directory Colors option is:

```
ext ... :ColorName; ...
```

where "ext" is either a file extension (which may include wildcards), 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

or a range (size, date, time, description, owner, and/or exclusion), or a file subsystem type:

```
EXETYPE_WIN32GUI       Windows x86 GUI app
EXETYPE_WIN32CUI       Windows x86 console app
EXETYPE_WIN64GUI       Windows x64 GUI app
EXETYPE_WIN64CUI       Windows x64 console app
EXETYPE_DOS            DOS (16-bit) app (obsolete)
EXETYPE_POSIX          POSIX app (obsolete)
EXETYPE_EFI            EFI app
```

and "ColorName" is any valid color name (see Colors and Color Names for information on color names). Specifying a subsystem type will significantly slow down the directory display, as TCC-RT has to read the header of each file to find a match.

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=exe:red; c asm:bright cyan; rdonly:green on white
```

To display 32-bit console apps in bright green and 64-bit console apps in bright red:

```
set colordir=EXETYPE_WIN32CUI:bri green;EXETYPE_WIN64CUI:bri red
```

Extended wildcards 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.

Options:

/	Display directory names with a trailing \.
/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 for details.
	You can specify **/A:=** to display a dialog to help you set individual attributes.
/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, hidden files are also displayed.
/L	Colorize the display. See **Colors** (above) for details.
/N	Disables the specified options:

           j   Skip junctions

/O:...	Sort the files before processing. 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, instead of numerically when numeric substrings are included in the name or extension.

c	Sort by compression ratio
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 description
o	Sort by owner
r	Reverse the sort order for all options
s	Sort by size
t	Same as **d**
u	Unsorted
z	Same as **s**

/P[*n***]**   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. The /P option has an optional argument *n* that specifies the number of seconds to wait for a keystroke before continuing.

/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. If you do not specify a number, /S shows the file sizes (see /Z).

/T   Display the time and date for each directory. If you combine **/T** and /F, the time and date for each file will also be displayed.

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 and FAT32 volumes).
/T:c   Creation date and time.
/T:w   Last modification ("*write*") date and time (default).

/Z   Display the size of each file. /Z without a /F will display the subdirectory tree sizes (the size of the current directory and all of its subdirectories).

4.2.204 TRUE

Purpose: Returns a 1

Format: TRUE

Usage:

TRUE returns 0 and sets the ERRORLEVEL variable to 1.

4.2.205 TRUENAME

Purpose: Find the full, true path and file name for a file

Format: TRUENAME *file*

See also: The @TRUENAME 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.

A leading ~\ or ~/ will be interpreted as the current user's home directory.

Example:

Call TRUENAME to get the true pathname for a file:

```
[c:\] subst d: c:\util\test
[c:\] truename d:\test.exe
c:\util\test\test.exe
```

4.2.206 TS

Purpose: Reads line from STDIN, prefix a date/time stamp, and write the lines to STDOUT

Format: TS [/D /T "*format*"]

"*format*"   The date / time format

/D(ate)
/T(ime)

Usage:

TS is intended to be used in pipes, when you need to know when each line was received.

If you don't specify any options, TS defaults to /D /T.

Options:

/"..."   The optional *format* string. See @DATEFMT for details on *format* arguments.

/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).

4.2.207 TYPE

Purpose: Display the contents of the specified file(s)

Format: TYPE [/A:[[-][+]rhsadecijopt] /B /I"*text*" /L[0] /O:[-]acdeginorstuz /P /X /XS] [@*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 for details).

/A: (Attribute select)	/P(ause)
/B(ell)	/O(rder)
/I"text" (match description)	/X (hex)
/L(ine numbers)	/XS (hex w/spaces)

See also: HEAD, TAIL, LIST.

File Selection

Supports attribute switches, extended wildcards, ranges, multiple file names, and include lists.

Internet: Can be used with FTP and HTTP servers, e.g.

```
type "https://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 (*.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 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 is useful if you want to add line numbers to a file, for example:

```
type /l myfile > myfile.num
```

TYPE sets two internal variables:

%_type_files	The number of files displayed
%_type_errors	The number of errors

TYPE will recognize Unicode (UTF-16) files based on either a BOM or specific UTF-16 sequences at the beginning of the file. TYPE will recognize UTF-8 files based on either a BOM or UTF-8 extended characters within the first 2K of the file.

- **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 for additional details.

Options:

/A: Select only those files that have the specified attribute(s) set. See Attribute Switches for information on the attributes which can follow **/A:**. Do not use /A: with @file lists. See @file lists for details.

You can specify **/A:=** to display a dialog to help you set individual attributes.

/B   Ignore bell (ASCII 7) characters.

/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 for details.

/L[*n***]**   Display a line number preceding each line of text. /L0 will not number blank lines.

/O:...   Sort the files before processing.

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, instead of numerically when numeric substrings are included in the name or extension.
c	Sort by compression ratio
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 description
o	Sort by owner
r	Reverse the sort order for all options
s	Sort by size
t	Same as **d**
u	Unsorted
z	Same as **s**

/P   Prompt after displaying each page. Your options at the prompt are explained in detail under Page and File Prompts.

/X   Display the file in hex.

/XS   Display the file in hex, using spaces instead of periods for non-printable characters.

4.2.208 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(uiet)](#)   [/R(ead file)](#)

See also: [ALIAS](#) and [ESET](#).

Usage:

TCC-RT 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.

You can use regular expressions in the alias name.

Examples:

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](#) 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](#) 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 if no filename is present and input is redirected.

4.2.209 UNBZIP2

Purpose: Uncompress bzip2 archives

Format: UNBZIP2 [/C /E /O /Q /V] *bziparchive* [*path*]

bzip2archive	The .bz2 file to work with
path	The path where files will be extracted

[/C(ontents)](#)   [/Q(uiet)](#)
[/E(xtract)](#)   [/V(iew)](#)
[/O(verwrite)](#)

Usage:

The UNBZIP2 command will uncompress archives that have been compressed using the bzip2 format.

Options:

/C   Display (on standard output) the contents of a file in the bzip2 archive.

/E   Extract (default).

/O   Overwrite existing files.

/Q   Don't display the filenames as they are extracted from the archive.

/V   View the list of files in the zip file (date, time, and filename).

4.2.210 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(uiet)](#)   [/R(ead file)](#)

See also: [FUNCTION](#) and [ESET](#).

Usage:

TCC-RT 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.

You can use regular expressions in the function name.

Examples:

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](#) 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](#) 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 if no filename is present and input is redirected.

4.2.211 UNGZIP

Purpose: Add, update, or delete files in a .gz (GZIP) archive

Format: UNGZIP [/A:[[-][+]rhsdaecjot] /E /O /Q /V] [*gziparchive*] *path*

gziparchive	The gzip file to work with
path	The path where files will be extracted

[/A:... (attribute switch)](#)   [/Q(uiet)](#)
[/E(xtract)](#)   [/V(iew)](#)
[/O(verwrite)](#)

See also GZIP.

File Selection

Supports attribute switches, extended wildcards, ranges, multiple file names, and include lists.

Usage:

UNGZIP is compatible with the archives created by the Linux / UNIX gunzip utility, and supports RFC 1952.

You can specify a pathname for *gziparchive*. If you don't provide an extension, and the filename as entered doesn't exist, GZIP adds ".gz". If you don't specify an operation, UNGZIP will default to Extract.

Option:

/A:...   Select only those files that have the specified attribute(s) set. See Attribute Switches for information on the attributes which can follow /A:. Do not use /A: with **@*file*** lists. See @file lists for details.

         You can specify **/A:=** to display a dialog to help you set individual attributes.

/E   Extract (default).

/O   Overwrite existing files.

/Q   Don't display the filenames as they are extracted from the archive.

/V   View the list of files in the zip file (date, time, and filename). Due to the limitations of the GZIP format, this can only display the first file in the archive. If the file was compressed with lzw, it will not have a header, so it cannot be viewed.

4.2.212 UNJAR

Purpose: Extract or list files in a Java JAR archive

Format: UNJAR [/A:[[-][+]rhsdaecjot] /C /E /F /O /P /Q /T /TEST /U /V] *jararchive* [*path*] [@*file*] *file*...

jararchive	The JAR file to work with
path	The path where files will be extracted
file	The file(s) to extract

/A:... (attribute switch)	/P(ercent)
/C(ontents)	/Q(uiet)
/E(xtract)	/TEST
/F(reshen)	/U(pdate)
/O(verwrite)	/V(iew)

Usage:

UNJAR will extract or list files in a Java JAR archive. The syntax is similar to the UNZIP command.

See also JAR.

Options:

/A:... Select only those files that have the specified attribute(s) set. See Attribute Switches for information on the attributes which can follow /A:. Do not use /A: with **@file** lists. See @file lists for details.

    You can specify **/A:=** to display a dialog to help you set individual attributes.

/C Display (on standard output) the contents of a file in the tar archive.

/E Extract the specified file(s). (This is the default.)

/F Extract only those files that currently exist in the target folder, and which are older than the file in the Jar archive.

/O Overwrite existing files. UNJAR normally prompts before overwriting an existing file; /O will suppress the prompt.

/P Display the progress (0 - 100%) for each file as it is extracted.

/Q Don't display filenames as they are extracted.

/TEST Test the integrity of the JAR file (header and contents). Any errors will be displayed on STDERR.

/U Extract files which either don't exist in the target folder, or which are older than the file in the Jar archive.

/V View the list of files in the archive (date, time, size, and filename).

4.2.213 UNLIBRARY

Purpose: Remove functions from the library function list

Format: UNLIBRARY [/Q /R *file*... (function ...)] *function*...
     or
    UNLIBRARY *

    function One or more library functions to remove from memory.
    file   One or more files from which to read library functions to be undefined.

    /Q(uiet)   /R(ead file)

See also: FUNCTION and ESET.

Usage:

TCC-RT maintains a list of the functions that you have defined. The UNLIBRARY command will remove functions from that list. UNLIBRARY supports wildcards in the function name.

Examples:

To remove the function DDIR:

```
unlibrary ddir
```

To remove all the functions:

```
unlibrary *
```

To remove all the functions that begin with "DD":

```
unlibrary dd*
```

You can delete all matching library 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*:

```
unlibrary (func1 func2) f*
```

If you keep functions in a file that can be loaded with the LIBRARY /R command, you can remove the functions by using the UNLIBRARY /R command with the same file name:

```
unlibrary /r function.lst
```

This is much faster than removing each function individually in a batch file, and can be more selective than using UNLIBRARY *.

Options:

/Q   Prevents UNLIBRARY from displaying an error message if one or more of the library functions do 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 LIBRARY /R command. You can use multiple files with one UNLIBRARY /R command by placing the names on the command line, separated by spaces:

```
unlibrary /r function1.lst function2.lst
```

4.2.214 UNMOUNTISO

Purpose: Unmount an ISO image previously mounted with MOUNTISO.

Format: UNMOUNTISO [*d:* | *d:\path*]

       d: Optional drive letter.
       d:\path   Optional mount path

See also MOUNTISO.

Usage:

UNMOUNTISO is only supported in Windows 8 or later.

You must be running an elevated session to unmount the ISO image.

4.2.215 UNMOUNTVHD

Purpose: Unmount a VHD or VHDX image previously mounted with MOUNTVHD

Format: UNMOUNTVHD [*d:* | *d:\path*]

    d: Optional drive letter.
    d:\path   Optional mount path

See also MOUNTVHD.

Usage:

You must be running an elevated session to unmount the VHD or VHDX image.

4.2.216 UNQLITE

Purpose: Create / Read / Write a NoSQL database

Format: UNQLITE [/RWC [/RO [/MM] /RW /TEMP /MM] [/DB:"name"] [/C] [/D key]
[/R key] [/KVBA "key" handle length] [/KVF "key" filename length]
[/KVFA "key" filename length] [/KVS "key" "value"] [/KVSA "key" "string"]

/C(lose)	/MM (memory mapped)
/D(elete)	/R(ead)
/DB:name (database name)	/RO (open read-only)
/KVB (create key/binary blob)	/RW (open read+write)
/KVBA (append key/binary blob)	/RWC (open read+write+create)
/KVF (create key/file)	/TEMP (temporary db)
/KVFA (append key/file)	
/KVS (create key/value)	
/KVSA (append)	

Usage:

UnQLite is an embedded NoSQL (Key/Value store and Document-store) database engine. UnQLite reads and writes directly to ordinary disk files. The complete database with multiple collections is contained in a single disk file. The database file format is cross-platform, you can copy a database between 32-bit and 64-bit systems or between big-endian and little-endian architectures.

You can only perform one operation (open, close, write, read, etc.) each time you call UNQLITE. For example:

```
unqlite  /db:"test.db"/rwc
unqlite /db:"test.db" /kvs "keyone" "This is a string value assigned to
keyone"
unqlite /db:"test.db" /c
```

If *filename* is ":mem:", then a private in-memory database is created. The in-memory database will be discarded when the database is closed.

UNQLITE does not support extended Unicode characters for the database name.

Both keys and values are treated as arrays of bytes, so the content can be ASCII strings, Unicode strings, binary blogs, or disk files.

The maximum size of a file for /KVF or /KVFA is dependent on the RAM and disk space available.

UNQLITE has an internal command variable:

%_unq_db   The name of the current database

Options:

/C	Close a database. If you omit the name, UNQLITE will close the most recently opened database.
/D	Delete the specified key
/DB:*name*	Open an existing database for a read / write / delete operation. The database name should be quoted. You need to use the same name when calling any of the read / write options. If this option is omitted, UNQLITE will use the last database name (if any).
/KVB	Create a key / binary blob value. If the key exists, it will be overwritten with the new value. *handle* is a handle returned by @BALLOC; *length* is the length to write (or -1 for the entire buffer).
/KVBA	Append a binary blob to the value of an existing key. *handle* is a handle returned by @BALLOC; *length* is the length to write (or -1 for the entire buffer).
/KVF	Create a key / file value pair. If the key exists, it will be overwritten with the new value. *length* is the length of the file to write (or -1 for the entire file).
/KVFA	Append a file to the value of an existing key. *length* is the length of the file to write (or -1 for the entire file).
/KVS	Create a key / value pair. If the key exists, it will be overwritten with the new value.
/KVSA	Append a string to the value of an existing key.
/MM	A read-only memory-mapped view of the database. Only valid when used with /RO.
/R	Read the specified key and display the value. If the key doesn't exist (or doesn't have a value) UNQLITE will not display anything.
/RO	Open the database in read-only mode. If the database does not exist, an error is returned.
/RW	Open the database with read+write privileges. If the database does not exist, an error is returned.

/RWC	Open a database with read+write privileges. The database is created if it doesn't exist.
/TEMP	A private, temporary on-disk database will be created. The database will be deleted when the database is closed.

4.2.217 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 remove (wildcards accepted except for registry variables).
file	One or more files from which to read variables to be removed.

/D(efault)	/S(ystem)
/E(nvironment)	/U(ser)
/Q(uiet)	/V(olatile)
/R(ead)	

See also: ESET and SET.

Usage:

UNSET removes one or more variables from the environment or from the Windows Registry.

You can also use regular expressions in the variable name.

UNSET can be used in a batch file, in conjunction with the SETLOCAL and ENDLOCAL commands, to clear the environment of variables that may cause problems for applications run from that batch file.

For more information on environment variables, see the SET command and the general discussion of the environment.

Note: You cannot use UNSET with GOSUB variables.

Use caution when removing environment variables, and especially when using **UNSET ∗**. Many programs will not work properly without certain environment variables; for example, TCC-RT depends on PATH.

Registry Variables: Default, System, User, and Volatile registry variables can be manipulated with the UNSET command's /D, /S, /U and /V switches, respectively. To remove the variable from both the registry and from the local environment, use both the /E 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.

Examples:

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*

Options:

/D   Delete a default variable from the registry (HKU\.DEFAULT\Environment).

/E   When used together with one of /D, /S, /U, or /V, unsets both the registry variable and the local environment variable.

/Q   Prevents UNSET from displaying any error messages.

/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 /R command (see SET for more details), or it could just be one variable per line, wildcards not processed.

            UNSET /R will read from STDIN 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)

4.2.218 UNSETARRAY

Purpose: Remove array variables

Format: UNSETARRAY [/Q] *name* [*name*...]
                    or
                UNSETARRAY *****

            name   One or more array variables to remove (wildcards accepted).

            /Q(uiet)

See also: SETARRAY.

Usage:

UNSETARRAY removes one or more array variables.

For more information on array variables, see the [SETARRAY](#) command.

Examples:

To remove the array variable **ARRAY1**:

```
unsetarray array1
```

If you use the command **UNSETARRAY ***, all of the array variables will be deleted:

```
unsetarray *
```

You can delete all matching array variables except for those specified by enclosing the exceptions in parentheses. For example, to remove all array variables beginning with "v" except for *var1* and *var2*:

```
unsetarray (var1 var2) v*
```

Options:

/Q   Prevents UNSETARRAY from displaying an error message if one or more of the array variables do not exist. This option is most useful in batch files, for removing a group of arrays when some of the arrays may not have been defined.

4.2.219 UNSETP

Purpose: Delete an environment variable in another process

Format: UNSETP *pid* [/R *filename*][(*except...*)] var

        pid   Process ID, or the window title, or the task name
        var   The variable name to delete. The name can contain wildcards

        /R   Read variables and values from a file

See also [SETP](#).

Usage:

You can delete all matching variables except for those specified by enclosing the exceptions in parentheses.

UNSETP works by injecting a dll into the specified process and executing a command in that dll to remove the environment variable. Depending on your Windows configuration, you may need to be running an elevated session for UNSETP to work.

Examples:

To remove all variables beginning with "v" except for *var1* and *var2* in the process with a PID of 1234:

```
unsetp 1234 (var1 var2) v*
```

4.2.220 UNTAR

Purpose:	Extract files from .TAR archives

Format: UNTAR [/A:[[-][+]rhsdaecjot] /C /D /E /F /G /Net /O /P /Q /TEST /U /V] *tararchive path file* ...

tararchive	The .tar file to work with
path	The path where files will be extracted
file	The file(s) to extract

[/A:... (attribute switch)](#)	[/O(verwrite)](#)
[/C(ontents)](#)	[/P(ercent)](#)
[/D(irectory)](#)	[/Q(uiet)](#)
[/E(xtract)](#)	[/TEST](#)
[/F(reshen)](#)	[/U(pdate)](#)
[/G(zip)](#)	[/V(iew)](#)
[/N (defaults)](#)	

See also [TAR](#).

File Selection

Supports [attribute switches](#), extended [wildcards](#), [ranges](#), [multiple file names](#), and [include lists](#).

Usage:

UNTAR is compatible with tar archives created in Linux / UNIX. Unlike .zip archives, .tar archives are not compressed unless you use the gzip option.

You can specify a pathname for *tararchive*. If you don't provide an extension, and the filename as entered doesn't exist, UNTAR adds ".tar". If you don't specify an operation, UNTAR will default to Extract.

UNTAR supports wildcards for the tar archive name and for the filenames to extract.

path specifies the path where files will be extracted. If *path* is not specified, files are extracted to the current directory.

UNTAR supports [gzip](#) decompression, and can be used to extract .tar.gz archives.

UNTAR sets two internal variables:

%_untar_files	The number of files extracted
%_untar_errors	The number of errors

Options:

/A:...	Select only those files that have the specified attribute(s) set. See [Attribute Switches](#) for information on the attributes which can follow /A:. Do not use /A: with **@*file*** lists. See [@file lists](#) for details.
	You can specify **/A:=** to display a dialog to help you set individual attributes.
/C	Display (on standard output) the contents of a file in the tar archive.

/D	Recreate the directory structure saved in the tar file.
/E	Extract the specified file(s). (This is the default.)
/F	Extract only those files that currently exist in the target folder, and which are older than the file in the tar archive.
/G	Use Gzip decompression.
/N	Disable one or more default behaviors:

> **e** Don't display errors.
> **t** Don't update the CD / CDD extended directory search database *(JPSTREE.IDX)*.

/O	Overwrite existing files. UNTAR normally prompts before overwriting an existing file; /O will suppress the prompt.
/P	Display the progress (0 - 100%) for each file as it is extracted.
/Q	Don't display filenames as they are extracted.
/TEST	Test the integrity of the TAR file (header and contents). Any errors will be displayed on STDERR.
/U	Extract files which either don't exist in the target folder, or which are older than the file in the tar archive.
/V	View the list of files in the archive (date, time, size, and filename).

4.2.221 UNZIP

Purpose: Extract files from .ZIP archives

Format: UNZIP [/A:[[-][+]rhsdaecjot] /C /CRC /D /E /F /I /Nt /O /P /Q /S"*password*" /TEST /U /V] *ziparchive path file* ...

ziparchive	The Zip file to work with
path	The path where files will be extracted
file	The file(s) to extract

/A:... (attribute switch)	/O(verwrite)
/C(ontents)	/P(ercent)
/CRC	/Q(uiet)
/D(irectory)	/S"password"
/E(xtract)	/TEST
/F(reshen)	/U(pdate)
/I (descriptions)	/V(iew)
/N	

File Selection

Supports attribute switches, extended wildcards, ranges, multiple file names, and include lists.

Usage:

You can specify a pathname for *ziparchive*. If you don't provide an extension, and the filename as entered doesn't exist, UNZIP adds ".zip". If you don't specify an operation, UNZIP will default to Extract.

UNZIP supports wildcards for the zip archive name and for the filenames to extract. UNZIP will prompt before overwriting existing files. Your options at the prompt are explained in detail under Page and File Prompts.

path specifies the path where files will be extracted. If *path* is not specified, files are extracted to the current directory.

UNZIP will automatically use the Zip64 extensions if the archive is in Zip64 format.

UNZIP sets two internal variables:

%_unzip_files	The number of files extracted
%_unzip_errors	The number of errors

Option:

/A:...   Select only those files that have the specified attribute(s) set. See Attribute Switches for information on the attributes which can follow /A:. Do not use /A: with ***@file*** lists. See @file lists for details.

You can specify ***/A:=*** to display a dialog to help you set individual attributes.

/C   Display (on standard output) the contents of a file in the zip archive.

/CRC   Display the file CRCs (must be used with /V).

/D   Recreate the directory structure saved in the zip file.

/E   Extract the specified file(s). (This is the default.)

/F   Extract only those files that currently exist in the target folder, and which are older than the file in the zip archive.

/I   Save the "File Comment" (if any) for each extracted file to the NTFS description or the DESCRIPT.ION file.

/Nt   Don't update the CD / CDD extended directory search database *(JPSTREE.IDX)*.

/O   Overwrite existing files. UNZIP normally prompts before overwriting an existing file; /O will suppress the prompt.

/P   Display the progress (0 - 100%) for each file as it is extracted.

/Q   Don't display filenames as they are extracted.

/S   Use the specified password to extract the file(s) from an encrypted archive. If you don't provide a password on the command line, UNZIP will prompt you to enter one.

/TEST   Test the integrity of the ZIP file (header and contents). Any errors will be displayed on STDERR.

/U   Extract files which either don't exist in the target folder, or which are older than the file in the zip archive.

/V   View the list of files in the archive (date, time, size, compression ratio, and filename). If the zip file is password protected, UNZIP will append a ***** after the filename.

4.2.222 UPTIME

Purpose: Display the time since startup and the active time

Format: UPTIME

Usage:

UPTIME displays the time since the system was last rebooted, and the time the system has been active (i.e., not sleeping or hibernating).

Example:

```
[D:\TakeCommand28]uptime

Uptime  4 days 14 hours 58 minutes 42 seconds
Boot    6/19/2021 3:31:35 AM
Logon   6/19/2021 10:23:51 AM
```

4.2.223 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)](#)

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 the last argument on the line is a single **(,** it is interpreted as the beginning of a command group. USBMONITOR will append the following lines (in a batch file) or prompt you for more input (at the command line) until it gets a closing **).**

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-RT thread is also performing I/O. You may need to use START or DETACH 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 (this will usually have special characters like **&** in the name, so you will probably need to use double quotes around the variable name to prevent TCC-RT from parsing the special characters)
_usbname	The "friendly" name of the device

There is another variable that is updated after each trigger:

_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.

4.2.224 UUID

Purpose:	Create UUIDs in different formats
Format:	UUID [/B] [/C*n*] [/F*n*]

> /B(races)
> /Cn (create)
> /Fn (format)

Usage:

A UUID (also referred to as a GUID in Windows) is a 128-bit integer number used to uniquely identify resources.

Examples:

```
[D:\TakeCommand28]uuid /b
{0634f6db-823e-4536-bb40-eb025d3020bc}

[D:\TakeCommand28]uuid /f1
5ED9BCD0-B6B0-4C01-822B-20A03CECB6B0

[D:\TakeCommand28]uuid /f3
42D9909D2385495D9D51C8C6F4AB8D36
```

Option:

/B	Enclose the UID in curly braces
/C*n*	Create *n* UUIDs

/F*n*   Format for the UUID, where *n* is:

> 0 - returns the UUID with lower case alphabetic characters and embedded hyphens
> 1 - returns the UUID with upper case alphabetic characters and embedded hyphens
> 2 - returns the UUID with lower case alphabetic characters and no hyphens
> 3 - returns the UUID with upper chase alphabetic characters and no hyphens

4.2.225 VBEEP

Purpose: Flash the screen and beep the speaker

Format: VBEEP [*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.

Usage:

VBEEP flashes the screen (by setting all attributes to their inverse), and 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.

You can include as many frequency and duration pairs as you wish. No sound will be generated for frequencies less than 20 Hz.. The default value for *frequency* is 440 Hz; the default value for *duration* is 2.

Numeric input may be entered in either decimal format (a sequence of 0-9 digits) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

4.2.226 VDESKTOP

Purpose: Manage Windows 10 virtual desktops

Format: VDESKTOP [[/N="*name"]* /C [/W="file"] /R id /S [id] - +]

/C(reate)
/R(emove)
/S(witch)
/N="name"
/W="file"

Usage:

VDESKTOP lets you create, remove, or switch Windows 10 virtual desktops.

VDESKTOP requires Windows 10 build 21313 or later.

Options:

/C	Create a new desktop
/N	You can optionally specify a desktop name. If you don't specify a name, you need to use a desktop number (1 - *n*) or the desktop GUID. Note that with the current Windows builds, the name is not updated in the Task View, though it is usable with subsequent VDESKTOP commands, and it will be displayed properly when the system is restarted.
/R	Remove the specified desktop. Id can be a desktop number (1 - n) or the GUID for that desktop.
/S	Switch to the specified desktop. If *id* isn't specified, switch to the desktop created with /C. *id* can either be a desktop number (1 - n) or the GUID for that desktop.
/W	When used with /C, /W specifies the image file to use for the background wallpaper for the desktop. Note that with the current Windows builds, the background will not be updated until the system is restarted.

4.2.227 VER

Purpose: Display the TCC-RT and operating system versions

Format: VER [/C /R]

/C(MD format)
[/R(evision)](#)

Usage:

Version numbers consist of a one or two-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-RT and Windows:

```
[c:\] ver
TCC  28.00.2 x64   Windows 10 [Version 10.0.21390.2025]
```

Option:

/C	Display Windows version information in the same format as CMD.EXE (i.e., "Microsoft Windows [Version 10.0.19559.1000]").
/R	Display the TCC-RT and operating system internal revision level (if any), plus your registered name.

4.2.228 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-RT supports VERIFY as a "do-nothing" command, for compatibility with CMD. 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
```

4.2.229 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.

Example:

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.

4.2.230 VSCRPUT

Purpose: Display text vertically in the specified color

Format: VSCRPUT *row col* [/C /U] [BRIght] *fg* ON [BRIght] *bg text*

row	Starting row
col	Starting column
fg	Foreground text color
bg	Background text color

text The text to display

 /C (move cursor)
 /U (move to end of string)

See also: [SCRPUT](#).

Usage:

VSCRPUT writes text vertically on the screen rather than horizontally. It can be used for simple graphs and charts generated by batch files.
Like the SCRPUT command, it uses the colors you specify to write the text. See [Colors and Color Names](#) 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-RT window. The maximum ***column*** value is determined by the current virtual screen width.

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***.

Example:

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
```

Options:

 /C Move the cursor to the specified position after writing the string.

 /U Move the cursor to the end of the string.

4.2.231 WAITFOR

 Purpose: Wait for an app to exit, or optionally for the app to finish processing its initial input and wait for user input.

 Format: WAITFOR [/Exit=*n* /Idle=*n*] [PID | "*title*" | *exename*]

 [/Exit](#) (ms to wait)

/Idle (ms to wait)

/PID (process ID)
title (window title)
exename

Usage:

You can choose the process by either passing the process ID, the window title, or the executable name.

The process ID can be either hex or decimal; if it is hex you must prefix it with **0x**.

The window title must be enclosed in double quotes; wildcards are supported.

Few apps will require the /Idle option.

Example:

Options:

/Exit - Wait for a maximum of *n* milliseconds for the process to exit. *n* will default to 10000ms (10 seconds).

/Idle - Wait for a maximum of *n* milliseconds for the process to enter the idle state. *n* will default to 10000ms (10 seconds).

4.2.232 WAKEONLAN

Purpose: Sends a "Wake-On_LAN" packet to a system

Format: WAKEONLAN *remotehost macaddress*

remotehost	The IP address of the machine to wake
macaddress	The physical address of the remote host

Usage:

WAKEONLAN send a "Wake-On-LAN" packet to the specified system (which may also be a broadcast address). This will power on the remote machine if the functionality is supported by the network card on the remote machine.

4.2.233 WATCH

Purpose: Run command(s) repeatedly, displaying the output and highlighting the difference(s) from the last run.

Format: WATCH [/A /B /C /D /F /H*n* /I*n* /M*n* /N*f* /N*h* /R"regex" /T*n* /U /V /X] "*command ...*"

/A highlight all changes	/M maximum count
/B beep if return != 0	/N[fh] no footer/header
/C clear screen	/R regex
/D disable colorization	/T footer lines

[/F prompt on change](#)   [/U beep if output changed](#)
[/H header lines](#)   [/V verbose](#)
[/I interval](#)   [/X exit if output changed](#)

Usage:

WATCH allows you to see how program output changes over time. WATCH will highlight the changed output text. To end the WATCH, press Esc or ^C.

Command can be an internal command, alias, batch file, or an external application.

Note that you need to double your variable %'s if you want the variables to be expanded by the specified commands instead of by WATCH.

Example:

```
watch /c /v "(echo time=%%_time & echo date=%%_date)"
```

Options:

/A	Highlights all changes between the current run and the first one,instead of the difference between the current and previous runs.
/B	Beeps if the return code != 0
/C	Clear the display and home the cursor before each run
/D	Disable the highlight colorization
/F	Freezes the display if the output changed, and prompts you to enter a key to continue.
/H*n*	Display only the leading *n* lines
/I*n*	Interval (in seconds) between each run
/M*n*	Maximum number of times to run the command
/N[fh]	Don't display the WATCH footer and/or footer
/R"*regex***"**	Only display the output line(s) that match the regular expression
/T*n*	Display only the trailing *n* lines
/U	Beep if the output changes
/V	Verbose output (header and footer)
/X	Exits if the output changes
"*command...***"**	Command(s) to execute

4.2.234 WEBFORM

Purpose: POST data to interactive web pages or scripts

Format: WEBFORM
[/A*n* /E*n* /F*n* /IPv6 /U"*username*" /O:*headers*" /P"*password*" /R"*referer*" /T*n* /V] /W"*url*"
"*varname*" "*varvalue*" ...

        varname   Form variable
        varvalue   Form value

/An (authorization)	/P(assword)
/En (encoding)	/R(eferrer)
/F (HTTP agent)	/Tn (firewall)
/IPv6	/U(ser)
/L(ocal file)	/V(erbose)
/O(ther headers)	/W(eb URL)

Usage:

WEBFORM will POST data to interactive web pages or scripts (CGI, ASP, etc.), similar to what HTML forms do.

WEBFORM will use the proxy & firewall settings from TCMD.INI. WEBFORM will support either HTTP or HTTPS (SSL) connections.

Example:

```
webform /v /w"http://download.finance.yahoo.com/d/quotes.csv" "f",
"sl1d1t1c1ohgv" "e", ".csv" "s", "IBM"
```

Options:

    /An   Authorization scheme:

        0 - basic
        1 - digest
        2 - proprietary
        3 - none
        4 - NTLM
        5 - Negotiate

    /En   Encoding:

        0 (URLEncoding) This is the most common encoding for HTML form contents.
        1 (MultipartFormData) This is MIME encoding allowing transmission of binary data.
        2 (QueryString) This is an older form of encoding where the actual parameters are appended to the URL query string. (Generally not recommended because most servers limit the size of the URL to less than 1K or 2K).

    /F   Email address of the HTTP agent.

    /IPv6   Use IPv6 instead of IPv4.

/L"localfile"

Local file for downloading. If the file exists, it will be overwritten.

/O"headers"

Other headers. The headers must be of the format "header: value" as described in the HTTP specifications. Header lines should be separated by CR/LF (^r^n).

/P"password"

Password if authentication is to be used.

/R"referer"   The document referring the requested URL

/Tn   Firewall type:

0 - no firewall (default)
1 - Connect through a tunneling proxy. Port is set to 80.
2 - Connect through a SOCKS4 proxy. Port is set to 1080.
3 - Connect through a SOCKS5 proxy. Port is set to 1080.

/U"username"

User name if authentication is to be used

/V   Display retrieved document text

/W"url"   URL of web page

4.2.235 WEBSOCKET

Purpose: Establish a WebSocket connection to a server and send a string.

Format: WEBSOCKET [/V /Origin=*server* /User=*user* /Password=*password*]] *"ws:servername"* *string*

V(erbose)
/Origin HTTP header
/User (for auth)
/Password (for auth)
servername
string (text to send)

Usage:

The options are position dependent; they can only appear at the beginning of the command line in the order specified above.

Example:

Options:

/V(erbose) - Display status messages

/Origin - If specified, WEBSOCKET will include an Origin HTTP header in the connection request with the value provided. Servers may use this value to validate requests. Servers may reject requests depending on the value provided. A typical value that would be set is of the form "http://example.com".

/User - The user name if authentication is used

/Password - The password if authentication is used

servername - The WebSocket server to connect to. For example: "ws://echo.websocket.org"

string - The text to send to the server

4.2.236 WEBUPLOAD

Purpose: Upload files to RFC1867-compliant web servers

Format: WEBUPLOAD
                   [/A
                   n /E
                   n /F"*from*" /IPV6 /L"*file*" /O"*headers*" /U"*username*" /P"*password*" /R"*referer*" /V] /W"*url*
                   " [/V "*varname*" "*varvalue*"] "*filevar*" "*filename*" ...

varname	Form variable
varvalue	Form value
filevar	The file(s) to extract
filename	The file(s) to upload

/An (authorization)	/P(assword)
/En (encoding)	/R(eferrer)
/F (HTTP agent)	/Tn (firewall)
/IPV6	/U(ser)
/L(ocal file)	/V(erbose)
/O(ther headers)	/W(eb URL)

Usage:

WEBUPLOAD will use the proxy & firewall settings from TCMD.INI. WEBUPLOAD will support either HTTP or HTTPS (SSL) connections.

Options:

/An   Authorization scheme:

               0 - basic
               1 - digest
               2 - proprietary
               3 - none
               4 - NTLM
               5 - Negotiate

/En   Encoding:

               0 (URLEncoding) This is the most common encoding for HTML form contents.

---

1 (MultipartFormData) This is MIME encoding allowing transmission of binary data.
2 (QueryString) This is an older form of encoding where the actual parameters are appended to the URL query string. (Generally not recommended because most servers limit the size of the URL to less than 1K or 2K).

/F"from"   Email address of the HTTP agent.

/IPv6   Use IPv6 instead of IPv4.

/U"username"
        User name if authentication is to be used.

/P"password"
        Password if authentication is to be used.

/L"localfile"
        Local file to upload.

/O"headers"
        Other headers. The headers must be of the format "header: value" as described in the HTTP specifications. Header lines should be separated by CR/LF (^r^n).

/R"referer"
        The document referring the requested URL

/Tn   Firewall type:

        0 - no firewall (default)
        1 - Connect through a tunneling proxy. Port is set to 80.
        2 - Connect through a SOCKS4 proxy. Port is set to 1080.
        3 - Connect through a SOCKS5 proxy. Port is set to 1080.

/V   The following two arguments are a varname / varvalue pair.

/W"url" URL of web page

4.2.237 WHICH

Purpose: Display the command type and what it would execute

Format: WHICH [/A] *command* [*command* ...]

        command   One or more commands or files.

        [/A(ll)](#)

Usage:

WHICH displays information about internal and external commands, library functions, [Aliases](#) (including keystroke aliases), files, plugin variables, internal variables, variable functions, and user-defined variable functions. When a file matches an applicable [Executable Extension](#) or [Windows File Association](#), 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
```

You can use regular expressions in the alias name. A leading ***** will skip the alias test (i.e., if **dir** is an alias, ***dir** will return the internal command).

WHICH can also recognize Plugin commands, REXX files, EXTPROC files, and associated files.

Note: WHICH does not support wildcard specifications unless you use the /A 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.

See Executable Files and File Searches for details on the order in which various locations are searched.

See also: @SEARCH, ASSOC, FTYPE.

Option:

/A Display all matching names. (Normally WHICH only displays the first match.) Executable files will be displayed in the order they are found in the PATH.

4.2.238 WINDOW

Purpose: Minimize or maximize the current window, restore the default window size, or change the window title

Format: WINDOW [MAX | MIN | RESTORE | HIDE | TRAY | TOPMOST | NOTOPMOST | TOP | BOTTOM | DETACH | /POS=*left,top,width,height* | /SIZE=*rows,columns* | /TRANS=*n* | /FLASH=*type,count* | VDESKTOP=*id* | "*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
type	Type of window flash
count	Number of times to flash the window

/FLASH /SIZE (of screen buffer)
/POS(ition)

See also: ACTIVATE and TITLE.

Usage:

`WINDOW` is used to control the appearance and title of the current (TCC-RT) window.
Numeric input may be entered in either decimal format (a sequence of 0-9 digits) 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, TOPMOST, NOTOPMOST, TOP, BOTTOM, and /TRANS options will be sent to the main *Take Command* window, not the TCC-RT 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	Description

/POS=*left,top,width,height*	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 sized and positioned so that parts of it are beyond the physical area of the display. The **/** before the keyword is optional. */POS* is not supported in *Take Command* tab windows (use ACTIVATE instead).
	
	/POS accepts a * value for any of the arguments. If the value is *, WINDOW will use the existing position / width / height value. For example, to resize a window without moving it:
	
	    WINDOW /POS=*,*,1200,800
	
	To move a window without resizing it:
	
	    WINDOW /POS=200,400,*,*

/TRANS=n	Set the transparency level of the *Take Command* window. *n* is an value from 0 (invisible) to 255 (opaque).

/FLASH=*type,count*	Flash the TCC-RT or *Take Command* window. The arguments are:
	    type - type of flash; one or more of the following values:
	        0 - stop flashing
	        1 - flash the window caption
	        2 - flash the taskbar button
	        4 - flash continuously until WINDOW is called again with
	        the /FLASH type set to 0

	8 - flash continuously until the window comes to the foreground (cannot be used with 4) *count* - the number of times to flash the wndow
newtitle	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	Expands the window to its maximum size.
MIN	Reduces the window to an icon.
RESTORE	Returns the window to its default size and location.
HIDE	Makes the window invisible. Use **RESTORE** to make the window visible.
TRAY	Moves the window to the taskbar tray.
/SIZE=*rows,columns*	Specify the *TCC-RT* 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.
TOPMOST	Keeps the *Take Command* window on top of all other windows until it closes, or **NOTOPMOST** is used. (Only valid in a tab window.)
NOTOPMOST	Allows other windows to overlay the *Take Command* window (this is the normal state for most windows). (Only valid in a tab window.)
TOP	Moves the *Take Command* window to the top of the window order, above all other non-**TOPMOST** windows. (Only valid in a tab window.)
BOTTOM	Moves the *Take Command* window to the bottom of the window order. (Only valid in a tab window.)
DETACH	Detach TCC-RT from a *Take Command* tab window.
VDESKTOP=*id*	Move the window to another virtual desktop. *Id* can be either a desktop number (1-*n*), the GUID for that desktop, or the desktop name. See VDESKTOP for more details.

4.2.239 WINSTATION

Purpose: Show the window stations and desktops on your system

Format: WINSTATION [/C /R /S] *winsta\desktop* [*command*]

/C(reate)
/R(un command)
/S(witch desktop)

Usage:

A *window station* contains a clipboard and one or more desktops. When a window station is created, it is associated with the calling process and assigned to the current session.

WinSta0 (the interactive window station*)* is the only window station that can display a user interface or receive user input. All other window stations are non-interactive, and they cannot display a user interface or receive user input. You can create other window stations, but you cannot create and switch to a desktop on anything other than WinSta0.

Option:

/C   Create a new winstation and desktop

/R   Run the specified command on the winstation\desktop. If you do not specify a command, WINSTATION will run Windows Explorer

/S   Switch to the specified desktop. (You cannot switch to any winstation other than "WinSta0"; this is a Windows restriction.)

4.2.240 WMIQUERY

Purpose: Query the Windows Management Interface

Format: WMIQUERY [/A /B /C /H /L /Q /USER=*username* /PASSWORD=*password*] *namespace* "*query string*" [*index*]

namespace	The namespace to query
"query string"	WQL query string
index	Class instance

[/A(ll instances)](#)   [/L (no blank lines)](#)
[/B(lank)](#)   [/PASSWORD](#)
[/C(lasses)](#)   [/Q(uiet)](#)
[/H(eader)](#)   [/USER](#)

Usage:

You can use a single period **.** for **namespace** to default to **root\cimv2**.

WMIQUERY also supports remote queries. The namespace argument for remote servers will look like "\\remote-server\root\cimv2" (substitute your server name for "remote-server").

For more details on what is available, see the Microsoft WMI and WQL documentation:

[WMI Reference – Win32 apps | Microsoft Docs](#)

Examples:

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.

/L   Don't separate records with a CR/LF. (This is probably only useful when you are querying single-line records.)

/PASSWORD=*password* The password to use for remote queries.

/Q   Don't display the property name when displaying properties.

/USER=*name*   The user name to use for remote queries.

4.2.241 WMIRUN

Purpose: Run WMI methods on a local or remote machine.

Format:

WMIRUN /USER=*user* /PASSWORD=*password* /CLASS=*classname* /METHOD=*methodname networkresource command*

/USER=*username*
/PASSWORD=*password*
/CLASS=*classname*
/METHOD=*methodname*
networkresource
command

Usage:

You must be running in an elevated session.

Example:

This command terminates process 26568 on the local machine:

```
WMIRUN /method=Terminate /class=Win32_Process "\\.\root\CIMV2"
Win32_Process.Handle="26568"
```

Options:

/USER The user name to use for remote queries

/PASSWORD   The password to use for remote queries

/CLASS The WMI class name

/METHOD   The WMI method name

networkresource   WMI namespace. The namespace argument for remote servers will look something like "\\remote-server\root\cimv2" (substitute your server name for "remote-server").

       command   The command you want WMI to run.

4.2.242 WSETTINGS

    Purpose: Display a Windows settings dialog

    Format: WSETTINGS [/=] *dialogname* ...

             dialogname   The settings dialog name

See also WSHELL and WSHORTCUT.

Usage:

If you don't enter any arguments, WSETTINGS will display its command dialog. The command dialog allows you to select the desired dialog from a list.

The settings dialogs that are available will depend on your Windows version. The list below is for Windows 10.

Options:

    /=   Display the WSETTINGS command dialog to help you set the command line options. The /= option can be anywhere on the line; additional options will set the appropriate fields in the command dialog.

    Dialogname is one or more of the following:

```
About
AccessWorkOrSchool
Accounts
AccountInfo
Activation
ActivityHistory
AdvancedDisplay
AirplaneMode
AppDiagnostics
AppsAndFeatures
AppsForWebsites
AppVolume
Audio
AutomaticFileDownloads
AutoPlay
Background
BackgroundApps
Backup
BatterySaver
BatterySaverSettings
BatteryUsageByApp
Bluetooth
Broadcasting
```

```
Calendar
CallHistory
Camera
CellularNetwork
Clipboard
ClosedCaptions
ConnectedDevices
Colors
Contacts
DataUsage
DateAndTime
DefaultApps
DeliveryOptimization
DeviceEncryption
Dial-up
DirectAccess
Display
Documents
DownloadMaps
Encryption
Email
EmailAndAppAccounts
Encryption
Ethernet
EyeTracker
Family & other users
FeedbackAndDiagnostics
FindMyDevice
Fonts
ForDevelopers
GameBar
GameDVR
GameMode
GraphicsSettings
HighContrast
Holographic
InkingAndTyping
Keyboard
Location
Lock screen
Magnifier
ManageOptionalFeatures
ManageKnownNetworks
Messaging
Microphone
MobileHotspot
Motion
Mouse
```

```
MouseAndTouchpad
Multitasking
Narrator
NetworkStatus
NFCAndProximity
Nightlight
Notifications
NotificationsAndActions
OfflineMaps
OptionalFeatures
OtherDevices
OtherOptions
Pen
Personalization
Phone
PhoneCalls
Pictures
PowerAndSleep
PrintersAndScanners
Privacy
ProjectingToThisPC
ProximitySensor
Proxy
Radios
Recovery
RegionAndLanguage
RemoteDesktop
Screen rotation
Settings
SharedExperiences
SigninOptions
Sound
Speech
SpeechInkingAndTyping
Start
Storage
SyncSettings
TabletMode
Taskbar
Tasks
Themes
Touchpad
Troubleshoot
Typing
USB
VideoPlayback
Videos
VoiceActivation
```

```
VPN
Wheel
WiFi
WiFiCalling
WindowsDefender
WindowsHelloFace
WindowsHelloFingerprint
WindowsInsiderProgram
WindowsSecurity
WindowsUpdate
WindowsUpdateAdvancedOptions
WindowsUpdateCheckForUpdates
WindowsUpdateHistory
WindowsUpdateRestartOptions
YourInfo
YourPhone
```

4.2.243 WSHELL

Purpose: Open a Windows Explorer window in the specified location

Format: WSHELL [/T] *folder*

> **folder**   The folder name to open
>
> /T   Use File Explorer in **Take Command**

See also WSHORTCUT.

Usage:

The folder names available will depend on your Windows version. Windows 7 will have a subset; Windows 8 more, and Windows 10 most or all of them depending on your configuration. The list below is from Windows 10.

Option:

> **/T**   WSHELL will normally call Windows Explorer to display the folder. You can use /T to have the folder open in the **Take Command** File Explorer instead.

Folder is one or more of the following:

```
AccountPictures
AddNewProgramsFolder
AdministrativeTools
AppData
ApplicationShortcuts
AppsFolder
AppUpdatesFolder
Cache
```

```
CameraRoll
CDBurning
ChangeRemoveProgramsFolder
CommonAdminTools
CommonAppData
CommonDesktop
CommonDocuments
CommonDownloads
CommonMusic
CommonPictures
CommonPrograms
CommonRingtones
CommonStartMenu
CommonStartup
CommonTemplates
CommonVideo
ConflictFolder
ConnectionsFolder
Contacts
ControlPanelFolder
Cookies
Cookies\Low
CredentialManager
CryptoKeys
Desktop
DeviceMetadataStore
DocumentsLibrary
Downloads
dpapiKeys
Favorites
Fonts
Games
GameTasks
History
HomeGroupCurrentUserFolder
HomeGroupFolder
ImplicitAppShortcuts
InternetFolder
Libraries
Links
LocalAppData
LocalAppDataLow
MusicLibrary
MyComputerFolder
MyMusic
MyPictures
MyVideo
Nethood
```

NetworkPlacesFolder
OneDrive
OneDriveCameraRoll
OneDriveDocuments
OneDriveMusic
OneDrivePictures
Personal
PicturesLibrary
PrintersFolder
PrintHood
Profile
ProgramFiles
ProgramFilesCommon
ProgramFilesCommonX64
ProgramFilesCommonX86
ProgramFilesX64
ProgramFilesX86
Programs
Public
PublicAccountPictures
PublicGameTasks
PublicLibraries
QuickLaunch
Recent
RecordedTVLibrary
RecycleBinrFolder
ResourceDir
RingTones
RoamedTileImages
RoamingTiles
SavedGames
Screenshots
Searches
SearchHistoryFolder
SearchHomeFolder
SearchTemplatesFolder
SendTo
StartMenuStart Menu
Startup
SyncCenterFolder
SyncResultsFolder
SyncSetupFolder
System
SystemCertificates
SystemX86
Templates
ThisPCDesktopFolder
UsersFilesFolder

```
UserPinned
UserProfiles
UserProgramFiles
UserProgramFilesCommon
UsersLibrariesFolder
VideosLibrary
Windows
```

4.2.244 WSHORTCUT

Purpose:	Call a Windows Explorer shortcut
Format:	WSHORTCUT [/T] *shortcut*

	folder	The folder name to open
	[/T](#)	Use File Explorer in ***Take Command***

See also [WSHELL](#).

Usage:

The shortcuts available will depend on your Windows version. Windows 7 will have a subset; Windows 8 more, and Windows 10 most or all of them depending on your configuration. The list below is from Windows 10.

Option:

/T	WSHORTCUT will normally call Windows Explorer to invoke the shortcut. You can use /T to have the shortcut invoked by the ***Take Command*** File Explorer instead.

Shortcut is one or more of the following:

```
3dObjectsFolder
AddNetworkLocation
AdministrativeTools
Applications
AutoPlay
BackupAndRestore
BitLocker
BluetoothDevices
ColorManagement
CommandFolder
CommonPlacesFolder
ControlPanel
ControlPanelAllTasks
ControlPanelCategoryView
ControlPanelIconsView
CredentialManager
DateAndTime
DefaultPrograms
```

```
DesktopFolder
DeviceManager
DevicesAndPrinters
Display
DocumentsFolder
DownloadsFolder
EaseOfAccessCenter
Email
FamilySafety
Favorites
FileExplorerOptions
FileHistory
FolderOptions
FontSettings
FontsFolder
FrequentFolders
GamesExplorer
GetPrograms
HelpAndSupport
HomeGroupSettings
HomeGroupUsers
Hyper-VRemoteFileBrowsing
IndexingOptions
Infared
InstalledUpdates
InternetExplorerOptions
KeyboardProperties
LanguageSettings
Libraries
LocationInformation
LocationSettings
MediaServers
MouseProperties
MusicFolder
MyDocuments
Network
NetworkAndSharingCenter
NetworkConnectionsPCsettings
NetworkConnections
NetworkWorkGroup
NotificationAreaIcons
NVIDIAControlPanel
OfflineFiles
OneDrive
PenAndTouch
Personalization
PictureFfolder
PortableDevices
```

```
PowerOptions
PreviousVersionsResultsFolder
PrinthoodDelegateFolder
Printers
ProgramsAndFeatures
PublicFolder
QuickAccess
RecentPlaces
Recovery
RecycleBin
RegionAndLanguage
ReliabilityMonitor
RemoteAppAndDesktopConnections
RemoteAssistance
RemotePrinters
RemovableDrives
RemovableStorageDevices
ResultsFolder
Run
Search
SearchEverywhere
SearchFiles
SecurityAndMaintenance
SetProgramAccess
ShowDesktop
Sound
SpeechRecognition
StorageSpaces
SyncCenter
SyncSetupFolder
System
SystemIcons
SystemRestore
TabletPC
TaskbarAndNavigation
TaskView
TextToSpeech
ThisDevice
ThisPC
Troubleshooting
UserAccounts
UserAccounts(netplwiz)
UserPinned
UserProfile
VideosFolder
WebBrowser
WindowsDefender
WindowsMobilityCenter
```

```
        WindowsFeatures
        WindowsFirewall
        WindowsToGo
        WindowsUpdate
        WorkFolders
```

4.2.245 Y

Purpose: Copy standard input to standard output, and then copy the specified file(s) to standard output

Format: Y [/D /T /F"*format*"] *file* ...

     file   The file or list of files to send to standard output.

     /D   Prefix each line with the current date (yyyy-mm-dd)
     /T   Prefix each line with the current time (hh:mm:ss.ms)

     /F"*format*"   A custom time/date *format* string. See @DATEFMT for details on the *format* arguments.

See also: TEE, piping and redirection.

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.

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 for more information on pipes.

Examples:

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
```

4.2.246 ZIP

Purpose: Add, update, or delete files in a .ZIP archive

Format: ZIP [/A:[[-][+]rhsdaecjot] /A /C /CRC /D /E*n* /F /G /I /L*n* /M /O:[-]
adegnrstu /P /Q /R /S"password" /T /TEST /U /V /YC /Z"*text*"] *ziparchive* [@*file*] *file*...

ziparchive	The zip file to work with
file	The files(s) to be added to the zip file

[/A:... (attribute switch)](#)	[/O:... (sort order)](#)
[/A(dd)](#)	[/P(rogress)](#)
[/C(ontents)](#)	[/Q(uiet)](#)
[/CRC](#)	[/R(ecurse)](#)
[/D(elete)](#)	[/S"password"](#)
[/En (encryption type)](#)	[/T (save attributes](#)
[/F(reshen)](#)	[/TEST](#)
[/G(enerate keys)](#)	[/U(pdate)](#)
[/I (save descriptions)](#)	[/V(iew)](#)
[/Ln (compression level)](#)	[/YC (AES 256)](#)
[/M(ove)](#)	[/Z (comment)](#)

See also [UNZIP](#).

File Selection

Supports [attribute switches](#), extended [wildcards](#), [ranges](#), [multiple file names](#), and [include lists](#).

Usage:

You can specify a pathname for *ziparchive*. If you don't provide an extension, and the filename as entered doesn't exist, ZIP adds ".zip". If you don't specify an operation, ZIP will default to Add.

ZIP will automatically use the Zip64 extensions if the archive is in Zip64 format.

ZIP supports connected web folders. If an HTML file (i.e., with an .htm or .html extension) is zipped, ZIP 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 added to the ZIP archive.

ZIP sets two internal variables:

%_zip_files	The number of files archived
%_zip_errors	The number of errors

Options:

/A:...	Select only those files that have the specified attribute(s) set. See [Attribute Switches](#) for information on the attributes which can follow /A:. Do not use /A: with **@*file*** lists. See [@file lists](#) for details.
	You can specify **/A:=** to display a dialog to help you set individual attributes.
/A	Add the specified file(s) to the zip file. (This is the default.)
/C	Display (on standard output) the contents of a file in the zip archive.
/CRC	Display the file CRCs (must be used with /V).
/D	Delete the specified file(s) from the zip file.

/En Set the encryption type (0=default, 1=AES 128-bit, 2=AES 192-bit, 3=AES 256-bit).

/F Update only those files that currently exist in the zip file, and which are older than the files on disk.

/G Generate unique keys for each file encrypted. This setting controls the algorithm that generates AES cryptographic keys from the Password specified. For added security, a random *salt* value is generated, and a unique key will be generated for every unique combination of Password and salt. If /G is specified, a unique salt value and key will be generated for each file encrypted. If /G is not specified, a single salt value and key will be generated for all encrypted files in the archive.

/I Save file descriptions (from DESCRIPT.ION or the NTFS description) as the compressed file's "File Comment".

/Ln Set the compression level (0 - 6, where 0=no compression, and 6=maximum compression). The default is 4.

/M Delete the files from the disk after adding them to the zip file.

/O:... Sort the files before processing.

You may use any combination of the sorting options below. If multiple options are used, the files 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, instead of numerically when numeric substrings are included in the name or extension.
   d Sort by date and time (oldest first); also see **/T:acw**
   e Sort by extension
   g Group subdirectories first, then files
   r Reverse the sort order for all options
   s Sort by size
   t Same as **d**
   u Unsorted

/P Display the progress (0 - 100%) for each file as it is zipped.

/Q Don't display the files being zipped.

/R If the argument is a subdirectory, copy all of the files in that subdirectory and all of its subdirectories to the zip file.

/S Use the specified password to encrypt the file(s). If you don't provide a password on the command line, ZIP will prompt you to enter one.

/T Save the file attributes (they will be set when the file is extracted).

/TEST Test the integrity of the ZIP file (header and contents). Any errors will be displayed on STDERR.

/U	Update files which either don't exist in the zip, or which are older than the files on disk.
/V	View the list of files in the zip file (date, time, size, compression ratio, and filename). If the zip file is password protected, ZIP will append a ***** after the filename.
/YC	Use AES 256-bit encryption instead of the default Zip 2.0 encryption.
/Z"..."	Set the comment for the zip file.

4.2.247 ZIPSFX

Purpose: Create a ZIP-compatible self-extracting archive

Format: ZIPSFX [/A /B /C /D"*path*" /F"*file*" /I /Ln /M"*message*" /N /P"*xxx*" /R /S"*password*" /X64] *exearchive directory*...

exearchive	The self-extracting executable
directory	The directory to be compressed into the self-extracting executable

/A(dmin)	/M(essage)
/B(anner)	/N (silent)
/C(aption)	/P"xxx" (parameters)
/D (target directory)	/R(ecurse)
/F (execute after open)	/S"password"
/I(nstall)	/X64 (64-bit executable)
/Ln (compression level)	

File Selection

Supports extended wildcards and ranges,

Usage:

You can specify a pathname for **exearchive**. If you don't provide an extension, and the filename as entered doesn't exist, ZIPSFX adds ".exe".

ZIPSFX sets two internal variables:

%_zipsfx_files	The number of files archived
%_zipsfx_errors	The number of errors

Options:

/A	Require admin privileges to execute the created file.
/B	Banner text to display before beginning the self-extraction.
/C	Caption for the self-extractor dialogs.
/D	Target directory for the self-extractor.
/F	Optional name of the file to execute (open) after the archive is extracted. This must be a relative path to a file in **directory**. If this is set to ".", the folder in which the archive has been decompressed will open in Windows Explorer. If it is set to "" (an empty string), the extractor will close and take no action.

/I	Install to the specified directory, run the file (see /F), and then remove the extracted files.
/Ln	Set the compression level (0 - 6, where 0=no compression, and 6=maximum compression). The default is 4.
/M	Message to notify the user that the extraction has completed normally.
/N	Silent installation - hide the extraction progress bar and the "success" window when the archive executable is run.
/P	Optional parameters to pass to the file to be executed. Only valid with /F.
/R	If the argument is a subdirectory, copy all of the files in that subdirectory and all of its subdirectories to the zip file.
/S	Use the specified password to extract the file(s). If you don't provide a password on the command line, ZIPSFX will prompt you to enter one.
/X64	Create a 64-bit executable.

4.2.248 7UNZIP

Purpose:	Extract files from .7Z archives
Format:	7UNZIP [/A:[[-][+]rhsdaecjot] /C /CRC /D /E /F /Nt /P /O /Q /S"*password*" /TEST /U /V] *ziparchive path file* ...

ziparchive	The 7Zip file to work with
path	The path where files will be extracted
file	The file(s) to extract

/A:... (attribute switch)	/O(verwrite)
/C(ontents)	/P(ercent)
/CRC	/Q(uiet)
/D(irectory)	/S"password"
/E(xtract)	/TEST
/F(reshen)	/U(pdate)
/N	/V(iew)

File Selection

Supports attribute switches, extended wildcards, ranges, multiple file names, and include lists.

Usage:

You can specify a pathname for *ziparchive*. If you don't provide an extension, and the filename as entered doesn't exist, 7UNZIP adds ".7z". If you don't specify an operation, 7UNZIP will default to Extract.
7UNZIP supports wildcards for the 7Zip archive name and for the filenames to extract. 7UNZIP will prompt before overwriting existing files. Your options at the prompt are explained in detail under Page and File Prompts.

path specifies the path where files will be extracted. If *path* is not specified, files are extracted to the current directory.

7UNZIP sets two internal variables:

    %_7unzip_files   The number of files extracted
    %_7unzip_errors The number of errors

Options:

/A:...   Select only those files that have the specified attribute(s) set. See Attribute Switches for information on the attributes which can follow /A:. Do not use /A: with ***@file*** lists. See @file lists for details.

    You can specify **/A:=** to display a dialog to help you set individual attributes.

/C   Display (on standard output) the contents of a file in the 7Zip archive.

/CRC   Display the file CRCs (must be used with /V).

/D   Recreate the directory structure saved in the 7Zip file.

/E   Extract the specified file(s). (This is the default.)

/F   Extract only those files that currently exist in the target folder, and which are older than the file in the 7Zip archive.

/Nt   Don't update the CD / CDD extended directory search database *(JPSTREE.IDX).*

/O   Overwrite existing files. 7UNZIP normally prompts before overwriting an existing file; /O will suppress the prompt.

/P   Display the progress (0 - 100%) for each file as it is extracted.

/Q   Don't display filenames as they are extracted.

/S   Use the specified password to extract the file(s) from an encrypted archive. If you don't provide a password on the command line, 7UNZIP will prompt you to enter one.

/TEST   Test the integrity of the 7Zip file (header and contents). Any errors will be displayed on STDERR.

/U   Extract files which either don't exist in the target folder, or which are older than the file in the 7Zip archive.

/V   View the list of files in the archive (date, time, size, and filename). If the 7Zip file is password protected, 7UNZIP will append a * after the filename.

4.2.249 7ZIP

Purpose: Add, update, or delete files in a .7Z archive

Format: 7ZIP [/A:[[-][+]rhsdaecjot] /A /C /CRC /D /F /K*n* /L*n* /M /O:[-]
               adegnrstu /P /Q /R /S"*password*" /T /TEST /U /V] *ziparchive* [@*file*] *file*...

               ziparchive   The 7zip file to work with
               file   The files(s) to be added to the 7zip file

/A:... (attribute switch)	/P(rogress)
/A(dd)	/Q(uiet)
/C(ontents)	/R(ecurse)
/CRC	/S"password"
/D(elete)	/T (save attributes
/F(reshen)	/TEST
/Kn (compression method)	/U(pdate)
/Ln (compression level)	/V(iew)
/M(ove)	
/O:... (sort order)	

File Selection

Supports attribute switches, extended wildcards, ranges, multiple file names, and include lists.

Usage:

You can specify a pathname for *ziparchive*. If you don't provide an extension, and the filename as entered doesn't exist, 7ZIP adds ".7z". If you don't specify an operation, 7ZIP will default to Add.

7ZIP supports connected web folders. If an HTML file (i.e., with an .htm or .html extension) is zipped, 7ZIP 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 added to the 7ZIP archive.

7ZIP sets two internal variables:

%_7zip_files	The number of files archived
%_7zip_errors	The number of errors

Options:

/A:...   Select only those files that have the specified attribute(s) set. See Attribute Switches for information on the attributes which can follow /A:. Do not use /A: with **@file** lists. See @file lists for details.

You can specify **/A:=** to display a dialog to help you set individual attributes.

/A   Add the specified file(s) to the zip file. (This is the default.)

/C   Display (on standard output) the contents of a file in the zip archive.

/CRC   Display the file CRCs (must be used with /V).

/D   Delete the specified file(s) from the zip file.

/F   Update only those files that currently exist in the 7zip file, and which are older than the files on disk.

/Kn   Set the compression method.

      0   LZMA
      1   BZip2
      2   Delta

3	Copy (no compression)
4	Deflate
5	LZMA2 (default)

/Ln Set the compression level (1 - 5, where 1=minimum compression, and 5=maximum compression). The default is 2.

/M Delete the files from the disk after adding them to the 7zip file.

/O:... Sort the files before processing.

You may use any combination of the sorting options below. If multiple options are used, the files 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, instead of numerically when numeric substrings are included in the name or extension.
d	Sort by date and time (oldest first); also see **/T:acw**
e	Sort by extension
g	Group subdirectories first, then files
r	Reverse the sort order for all options
s	Sort by size
t	Same as **d**
u	Unsorted

/P Display the progress (0 - 100%) for each file as it is zipped.

/Q Don't display the files being zipped.

/R If the argument is a subdirectory, copy all of the files in that subdirectory and all of its subdirectories to the zip file.

/S Use the specified password to encrypt the file(s). If you don't provide a password on the command line, 7ZIP will prompt you to enter one.

/T Save the file attributes (they will be set when the file is extracted).

/TEST Test the integrity of the 7ZIP file (header and contents). Any errors will be displayed on STDERR.

/U Update files which either don't exist in the 7zip, or which are older than the files on disk.

/V View the list of files in the 7zip file (date, time, size, and filename). If the 7zip file is password protected, 7ZIP will append a ***** after the filename.

4.3   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]**. If you want to use a variable or expression for an array variable name, the syntax is **%[var[*n*]]**.

You can create, alter, view, and delete environment variables with the SET, ESET, and UNSET commands.

TCC-RT also supports two special types of variables:

▸ Internal variables are similar to environment variables, but are interpreted internally by TCC-RT, 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 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-RT, you can use the FUNCTION command to create your own. These latter ones are referred to as user defined variable functions or UDFs.

You can return the result of a command with %(*command*). This is the same as using the @EXEC variable function but a little easier to write.

%((...)) will evaluate and substitute the numeric expression. For example:

    echo %((3+5)) is the answer.

%[[...]] will evaluate the conditional expression, and return 0 if the exit status is true; 1 if it is not. For example:

    echo %[[5 == 6]]

Note: TCC-RT 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-RT session. Note that if the starting process's environment is changed (through registry modifications, for example) while TCC-RT is already running, those changes will not be automatically reflected in *TCC*'s current environment. See the SET command for details.

You use the SET command to create a new environment variable. SET can also modify or delete a single environment variable, or display the value of one or more environment variables. ESET allows you to edit an environment variable. UNSET 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*".

In addition, TCC-RT uses the environment to keep track of the default directory on each drive. Windows only tracks the default directory of the current drive; TCC-RT 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 command.

The trailing percent sign that was traditionally required for environment variable names is not usually required by TCC-RT, 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 with CMD.

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:

```
original         expanded        method
%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 for details.

Environment variables may contain alias names. TCC-RT 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 versions, TCC-RT 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.

4.3.1 Array Variables

In addition to environment variables, TCC-RT also supports up to 4-dimensional array variables. Array variables are defined by the SETARRAY command, and you assign values to them with the SET command.

See also UNSETARRAY, @ARRAYINFO, @EXECARRAY and @FILEARRAY.

4.3.2 CMD.EXE Variables

CMD 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).

CMDCMDLINE - the command line that started the command processor.

CMDEXTVERSION - the command extensions internal version number.

DATE - the current system date (see also _DATE).

RANDOM - a random number between 0 and 32767 (see also @RANDOM).

TIME - the current system time (see also _TIME).

TCC-RT supports all of these built-in variables. (In TCC-RT, **CMDEXTVERSION** will always return **2**.)

The variables below are used by some Microsoft command processors, but are ignored by TCC-RT. To see their usage by Microsoft and the alternate methods to achieve the same purpose in TCC-RT, review:

COPYCMD	CMD default options for COPY command
DIRCMD	CMD default options for DIR command

4.3.2.1 COPYCMD

The **COPYCMD** variable is used by some versions of CMD to hold default options for the COPY command. **TCC-RT** 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, 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.

4.3.2.2 DIRCMD

The **DIRCMD** variable is used by some versions of CMD to hold default options for the DIR command. *TCC-RT* 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, 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.

4.3.2.3 String substitution

For compatibility with some peculiar syntax introduced in recent CMD versions, TCC-RT 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.

4.3.3 Variables

Internal variables are special variables built into TCC-RT 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-RT, and cannot be changed with the SET, UNSET, ESET 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 will not return accurate results if your operating system and Advanced Power Management drivers do not provide correct information on battery status to TCC-RT.

For a list of internal variables organized by general categories of use, see Internal Variables by Category.

Examples

You can use internal variables in a wide variety of ways depending on your needs. Here are just a couple of examples:

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
```

4.3.3.1 Variables by Name

!	Last argument of the previous command
?	Exit code, last external program
_4ver	TCC-RT 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

_batch	Batch nesting level
_batchline	Line number in current batch file.
_batchname	Full path and filename of current batch file.
_batchpath	Pathname of the 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
cmdspec	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
consoleb	Handle to console screen buffer
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
elevated	1 if the TCC-RT process is elevated
errorlevel	Exit code, last external program
execarray	Array elements assigned by the last @EXECSTR

execstr	@EXECSTR return code
exit	*TCC-RT* exit return code
expansion	SETDOS /X value

fg	Foreground color at cursor position
filearray	Array elements returned by the last @FILEARRAY
ftperror	Last FTP error code

gpsalt	Altitude from sea level in meters
gpsazimuth	List of the azimuth of each satellite in view
gpselevation	List of the elevation of each satellite in view
gpserrorradius	Accuracy of the latitude and longitude values
gpsfixquality	Quality of the fix
gpsfixtype	Type of the fix
gpshdop	Horizontal dilution of precision
gpsheading	True heading
gpsids	List of the IDs of the satellites in view
gpslat	Latitude
gpslon	Longitude
gpsmagheading	Magnetic heading
gpsnmea	NMEA sentence
gpsopmode	GPS operation mode
gpspdop	Position dilution of precision
gpsprns	List of the PRN numbers of the satellites in view
gpssatsinview	Number of satellites in view
gpssatsused	Number of satellites used in solution
gpsselmode	GPS selection mode
gpssnr	List of the signal to noise ratio of each satellite in view
gpsspeed	Speed in knots
gpsstatus	GPS status
gpsvdop	Vertical dilution of precision

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
hyperv	Running inside Hyper-V: 1; else 0

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	IFTPS session active: 1, else 0
imonth	Current month name, local language, short
imonthf	Current month name, local language, full
ininame	Full pathname of the current INI file
insert	Current input editor state (0=overstrike, 1=insert)
ip	IP address(es) of local computer.
ipadapter	Index of the current adapter
ipadapters	Number of adapters in the system

iparpproxy	Returns 1 if the local computer is acting as an ARP proxy
ipdns	Returns 1 if DNS is enabled on the local computer
ipdnsother	List of other DNS servers configured for the host
ipdnsserver	The default DNS server for the local computer
iprouting	Returns 1 if routing is enabled on the local computer
ipv6	IP address(es) of local computer.
isftp	IFTP SSH session active: 1, else 0
isodate	Current date in ISO 8601 format
isodowi	ISO 8601 numeric day of week
isowdate	ISO 8601 current week date (yyyy-Www-d)
isoweek	ISO 8601 week of year
isowyear	ISO 8601 week date year

kbhit	Keyboard input character is waiting: 1, else 0

lalt	left Alt key depressed: 1, else 0
lastdir	Previous directory (from directory history)
lastdisk	Last valid drive
lctrl	Left Ctrl key depressed: 1, else 0
logfile	Current log file name
lshift	Left Shift key depressed: 1, else 0

minute	Current minute
monitors	Number of monitors
month	Current month of the year as integer
monthf	Current month of the year, English, full
msg_checkbox	If user has checked MSGBOX checkbox : 1, else 0

numlock	NumLock on: 1, else 0

openafs	OpenAFS installed: 1, otherwise 0
osbuild	Windows build number

parent	Name of the parent process
pbatchname	Name of the parent batch file
pid	TCC-RT process ID (numeric)
pipe	Current process is running in a pipe: 1, else 0
ppid	Process ID of parent process

ralt	Right Alt key depressed: 1, else 0
rctrl	Right Ctrl key depressed: 1, else 0
ready	List of accessible drives
registered	Registered user name
row	Current cursor row
rows	Screen height
rshift	Right Shift key depressed: 1, else 0

scrolllock	ScrollLock on: 1, else 0
second	Current second
selected	Selected text in the *TCC* window
serialports	Display available serial pors (COM1 - COMn)

service	TCC is a service: 1, else 0
shell	Shell level
shift	Shift key depressed: 1, else 0
shortcut	Pathname of shortcut that started this process
shralias	SHRALIAS is loaded: 1, else 0
startpath	Startup directory of current shell.
startpid	Process ID of most recent STARTed process
stdin	STDIN redirected: 0, else 1
stdout	STDOUT redirected: 0, else 1
stderr	STDERR redirected: 0, else 1
stzn	Name of time zone for standard time
stzo	Offset in minutes from UTC for standard time
syserr	Latest Windows error code

tccinstances	Number of current *TCC* instances
tccrt	Running inside TCC RT: 1, else 0
tccrun	Length of time TCC-RT session has been running
tccstart	Time TCC-RT session was started
tcexit	Pathname to TCEXIT.*
tcfilter	Current filter in the File Explorer window
tcfolder	Selected folder in the File Explorer window
tclistview	Selected entries in the File Explorer window
tcmdinstances	Number of current *Take Command* instances
tcstart	Pathname of the active TCSTART.*
tctab	Running inside Take Command: 1; else 0
tctabactive	*TCC* is active *Take Command* tab window: 1; else 0
tctabs	Current number of *Take Command* tab windows
time	Current time
transient	Current process is a transient shell: 1, else 0
tzn	Name of current time zone
tzo	Offset in minutes from UTC for current time zone

unicode	Shell uses unicode for redirected output: 1, else 0
utctime	Current UTC time
utcdate	Current UTC date
utcdatetime	Current UTC date and time
utchour	Current UTC hour
utcisodate	Current UTC date in ISO format
utcminute	Current UTC minute
utcsecond	Current UTC second

vermajor	TCC-RT major version
verminor	*TCC-RT* minor version
version	*TCC-RT* version in major.minor format (i.e., **25.0**)
virtualbox	Running inside VirtualBox: 1; else 0
virtualpc	Running inside VirtualPC: 1; else 0
vmware	Running inside VMWare: 1; else 0
vxpixels	Virtual screen horizontal size
vypixels	Virtual screen vertical size

windir	Windows directory pathname

winfgwindow	Title of foreground window.
winname	Name of local computer
winsysdir	Windows system directory pathname
winticks	Milliseconds since Windows was started
wintitle	Current window title
winuser	Name of current user.
winver	Windows version number
wow64	Running inside WOW64: 1; else 0
wow64dir	System WOW64 directory

x64	1 if TCC-RT is the x64 (64-bit) version
xen	Running inside Xen: 1; else 0
xmouse	Column of last mouse click
xpixels	Physical screen horizontal size in pixels
xwindow	Width of *Take Command* or TCC-RT window in pixels

year	Current year
ymouse	Row of last mouse click
ypixels	Physical screen vertical size in pixels
ywindow	Height of *Take Command* or TCC-RT window in pixels

4.3.3.2 Variables by Category

- ▸ TCC status
- ▸ Dates and times
- ▸ Drives and directories
- ▸ Error codes
- ▸ Hardware status
- ▸ Operating system and software status
- ▸ Screen, color, and cursor

The list below gives a one-line description of all Internal Variables 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.

Hardware status

acstatus	AC line status
alt	Alt key depressed
battery	Battery status
batterylife	Remaining battery life, seconds
batterypercent	Remaining battery life, %
capslock	CapsLock on: 1, otherwise 0
cpu	CPU type
cpuusage	CPU time usage (percent)
ctrl	Ctrl key depressed: 1, otherwise 0
kbhit	A keyboard input character is waiting: 1, otherwise 0
lalt	left Alt key depressed: 1, otherwise 0
lctrl	left Ctrl key depressed: 1, otherwise 0

lshift	left Shift key depressed: 1, otherwise 0
numlock	NumLock on:r 1, otherwise 0
ralt	right Alt key depressed: 1, otherwise 0
rctrl	right Ctrl key depressed: 1, otherwise 0
rshift	right Shift key depressed: 1, otherwise 0
scrolllock	ScrollLock on: 1, otherwise 0
shift	Shift key depressed: 1, otherwise 0

Operating system and software status

!	Last argument of previous command
admin	1 if administrator; else 0
boot	Boot drive letter, without a colon
codepage	Current code page number
country	Current country code
dos	Operating system type
dosver	Operating system version
elevated	1 if the TCC-RT process is elevated
host	Host name of local computer.
hwprofile	Windows hardware profile if defined
hyperv	Running inside Hyper-V: 1; else 0
idleticks	Milliseconds since last user input
ip	IP address(es) of local computer.
ipadapter	Index of the current adapter
ipadapters	Number of adapters in the system
iparpproxy	Returns 1 if the local computer is acting as an ARP proxy
ipdns	Returns 1 if DNS is enabled on the local computer
ipdnsother	List of other DNS servers configured for the host
ipdnsserver	The default DNS server for the local computer
iprouting	Returns 1 if routing is enabled on the local computer
ipv6	IP address(es) of local computer.
osbuild	Windows build number
serialports	Display available serial ports (COM1 - COMn)
tctab	Running inside Take Command: 1; else 0
virtualbox	Running inside VirtualBox: 1; else 0
virtualpc	Running inside VirtualPC: 1; else 0
vmware	Running inside VMWare: 1; else 0
windir	Windows directory pathname
winfgwindow	Title of foreground window.
winname	Name of local computer
winsysdir	Windows system directory pathname
winticks	Milliseconds since Windows was started
wintitle	Current window title
winuser	Name of current user.
winver	Windows version number
wow64	Running in Windows x64: 1; else 0
wow64dir	System WOW64 directory
x64	1 if TCC-RT is the x64 (64-bit) version
xen	Running inside Xen: 1; else 0

TCC-RT status

4ver	TCC-RT version
batch	Batch nesting level

batchline	Line number in current batch file.
batchname	Full path and filename of current batch file
batchpath	Pathname of the current batch file
batchtype	Type of the current batch file
bdebugger	Batch debugger active: 1, otherwise 0
build	Build number
childpid	Process ID of most recent child process
cmdline	Current command line
cmdproc	Command processor name
cmdspec	Full pathname of command processor
consoleb	Handle to the active console screen buffer
detachpid	Process ID of most recent detached process
dname	Name of the description file.
echo	Echo status
editmode	Default insert mode: 1; else 0
execarray	Array elements assigned by the last @EXECARRAY
exit	TCC-RT exit code
expansion	Current expansion mode (SETDOS /X)
filearray	Array elements assigned by the last @FILEARRAY
hlogfile	Current history log file name
iftp	IFTP session active: 1, otherwise 0
iftps	IFTPS session active: 1, otherwise 0
isftp	ISFTP (SSH) session active: 1, otherwise 0
ininame	Full pathname of the current INI file
insert	Current input editor state (0=overstrike, 1=insert)
logfile	Current log file name
msgbox_checkbox	Checked box in MSGBOX : 1, else 0
parent	Name of the parent process
pbatchname	Name of the parent batch file
pid	The TCC-RT process ID (numeric)
pipe	Current process is running in a pipe: 1, otherwise 0
ppid	Process ID of parent process
registered	Registered user name
selected	Selected text in current tab window
service	*TCC* is a service: 1, else 0
shell	Shell level
shortcut	Pathname of shortcut that started this process
shralias	SHRALIAS is loaded: 1, otherwise 0
startpath	Startup directory of current shell.
startpid	Process ID of most recent STARTed process
stdin	STDIN redirected: 0, otherwise 1
stdout	STDOUT redirected: 0, otherwise 1
stderr	STDERR redirected: 0, otherwise 1
tccinstances	Number of *TCC* instances
tccrt	Running inside TCC RT: 1, else
tccrun	Time current TCC-RT session has been running
tccstart	Time current *TCC-RT* session was started
tctabactive	*TCC* is the active **Take Command** tab: 1, else 0
tcexit	Current value of TCEXIT.*
tcfilter	Current filter in the File Explorer window
tcfolder	Selected folder in File Explorer window

tclistview	Selected entries in the File Explorer window
tcmdinstances	Number of *Take Command* instances
tcstart	Current value of TCSTART.*
tctabs	Current number of *Take Command* tab windows
transient	Current process is a transient shell: 1, otherwise 0
unicode	TCC-RT uses unicode for redirected output: 1, otherwise 0
vermajor	TCC-RT major version
verminor	TCC-RT minor version
version	*TCC-RT* version in major.minor format (i.e., **31.0**)

Screen, color, and cursor

bg	Background color at cursor position
ci	Current text cursor shape in insert mode
co	Current text cursor shape in overstrike mode
column	Current cursor column
columns	Virtual screen width
fg	Foreground color at cursor position
monitors	Number of monitors
row	Current cursor row
rows	Screen height
selected	Selected text in current tab window
vxpixels	Virtual screen horizontal size
vypixels	Virtual screen vertical size
xmouse	Column of last mouse click
xpixels	Physical screen horizontal size in pixels
xwindow	Width of *Take Command* or *TCC-RT* window in pixels
ymouse	Row of last mouse click
ypixels	Physical screen vertical size in pixels
ywindow	Height of *Take Command* or *TCC-RT* window in pixels

Drives and directories

afswcell	OpenAFS workstation cell
cdroms	List of CD-ROM drives
cwd	Current drive and directory
cwds	Current drive and directory with trailing \
cwp	Current directory
cwps	Current directory with trailing \
disk	Current drive
drives	List of all available drives
dvds	List of DVD drives
hdrives	List of hard (fixed) drives
lastdir	Previous directory (from directory history)
lastdisk	Last valid drive
openafs	OpenAFS service installed: 1, otherwise 0
ready	List of ready (accessible) drives

Dates and times

date	Current date
datetime	Current date and time, yyyyMMddhhmmss
day	Current day of the month
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
dst	Daylight savings time: 1, else 0
hour	Current hour
idow	Current day of the week, local language, short
idowf	Current day of the week, local language, full
imonth	Current month name, local language, short
imonthf	Current month name, local language, full
isodate	Current date in ISO 8601 format
isodowi	ISO 8601 numeric day of week
isowdate	ISO 8601 current week date (yyyy-Www-d)
isoweek	ISO 8601 week of year
isowyear	ISO 8601 week date year
minute	Current minute
month	Current month of the year as integer
monthf	Current month of the year, English, full
second	Current second
stzn	Name of time zone for standard time
stzo	Offset in minutes from UTC for standard time
time	Current time
tzn	Name of current time zone
tzo	Offset in minutes from UTC for current time zone
utctime	Current UTC time
utcdate	Current UTC date
utcdatetime	Current UTC date and time
utchour	Current UTC hour
utcisodate	Current UTC date in ISO format
utcminute	Current UTC minute
utcsecond	Current UTC second
year	Current year

Error codes

?	Exit code, last external program
_?	Exit code, last internal command
errorlevel	Exit code, last external program
_execstr	Last @EXECSTR return code
_ftperror	Last FTP error code
_syserr	Latest Windows error code

Windows Sensor

gpsalt	Altitude from sea level in meters
gpsazimuth	List of the azimuth of each satellite in view
gpselevation	List of the elevation of each satellite in view
gpserrorradius	Accuracy of the latitude and longitude values
gpsfixquality	Quality of the fix
gpsfixtype	Type of the fix
gpshdop	Horizontal dilution of precision
gpsheading	True heading
gpsids	List of the IDs of the satellites in view
gpslat	Latitude
gpslon	Longitude
gpsmagheading	Magnetic heading

gpsnmea	NMEA sentence
gpsopmode	GPS operation mode
gpspdop	Position dilution of precision
gpsprns	List of the PRN numbers of the satellites in view
gpssatinview	Number of satellites in view
gpssatsused	Number of satellites used in solution
gpsselmode	GPS selection mode
gpssnr	List of the signal to noise ratio of each satellite in view
gpsspeed	Speed in knots
gpsstatus	GPS status
gpsvdop	Vertical dilution of precision

4.3.3.3   Command Variables

Most of the TCC-RT file handling commands set internal variables with their results (for example, the number of files processed and the number of errors). See the help for the specific command for more details on the variables.

    _7unzip_files
    _7unzip_errors
    _7zip_files
    _7zip_errors
    _attrib_dirs
    _attrib_errors
    _attrib_files
    _copy_dirs
    _copy_errors
    _copy_files
    _dedupe_errors
    _dedupe_files
    _del_dirs
    _del_errors
    _del_files
    _differ_added
    _differ_changed
    _differ_deleted
    _differ_errors
    _dir_dirs
    _dir_errors
    _dir_files
    _do_dirs
    _do_errors
    _do_files
    _do_loop
    _ffind_errors
    _ffind_files
    _ffind_matches
    _foldertime
    _for_errors
    _for_files
    _head_errors
    _head_files
    _md_dirs
    _md_errors

_mklink_errors
_mklink_links
_mklnk_errors
_mklnk_links
_move_dirs
_move_errors
_move_files
_pdir_dirs
_pdir_errors
_pdir_files
_rd_dirs
_rd_errors
_ren_dirs
_ren_errors
_ren_files
_sync_dirs
_sync_errors
_sync_files
_tail_errors
_tail_files
_tar_errors
_tar_files
_touch_dirs
_touch_errors
_touch_files
_type_errors
_type_files
_untar_errors
_untar_files
_unzip_errors
_unzip_files
_zip_errors
_zip_files
_zipsfx_errors
_zipsfx_files

4.3.3.4   ! (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.

4.3.3.5   ? 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 - to emulate Microsoft's CMD - also set this variable to the same value they set the variable _?, 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 _? 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.

See also: _?

4.3.3.6 _? 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 TCC-RT 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 command.

Note that in imitation of CMD some internal commands, e.g., DIR, also set the variables ? and ERRORLEVEL to the same value they set this variable. However, you are strongly urged to use this variable.

See also: ?

4.3.3.7 _4VER

_4VER returns the current TCC-RT version (for example, 32).

See also: _BUILD.

4.3.3.8 _ACSTATUS

_ACSTATUS returns the AC line status.

value	meaning
0	Offline
1	Online
unknown	Unknown

4.3.3.9 _ADMIN

_ADMIN returns 1 if the current user is an administrator in the local group.

See also _ELEVATED.

4.3.3.10 _AFSWCELL

_AFSWCELL returns the OpenAFS workstation cell.

See http://www.openafs.org for more information on OpenAFS.

4.3.3.11 _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

4.3.3.12 _BATCH

_BATCH returns the current batch file nesting level. It is **0** if no batch file is currently being processed.

Batch files are nested with the internal CALL command.

4.3.3.13 _BATCHLABEL

_BATCHLABEL returns the name of the current GOSUB subroutine. (Or an empty string if not in a subroutine.)

4.3.3.14 _BATCHLINE

_BATCHLINE returns the current line number in the current batch file. It is **-1** if no batch file is active.

The first line in the batch file is numbered **1**.

4.3.3.15 _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.

4.3.3.16 _BATCHPATH

_BATCHPATH returns the pathname of the current batch file (including the trailing \).

4.3.3.17 _BATCHTYPE

_BATCHTYPE returns the file type of the current batch file:

value	meaning
-1	not in a batch file
0	normal
1	compressed
2	encrypted

4.3.3.18 _BATTERY

_BATTERY returns the battery charge status:

value	meaning
1	High
2	Low
4	Critical
8	Charging
128	No battery
unknown	Unknown

4.3.3.19 _BATTERYLIFE

_BATTERYLIFE returns either the number of seconds of battery life remaining, or **unknown**.

4.3.3.20 _BATTERYPERCENT

_BATTERYPERCENT returns the percentage of battery charge remaining (**0...100**), or **unknown**.

4.3.3.21 _BDEBUGGER

_BDEBUGGER returns **1** if the batch debugger is actively debugging a file, or **0** if it is not.

4.3.3.22 _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 for details.

4.3.3.23 _BOOT

_BOOT returns the boot drive letter, without a colon.

4.3.3.24 _BUILD

_BUILD returns the internal TCC-RT build number.

See also: _4VER.

4.3.3.25 _BTDEVICECOUNT

_BTDEVICECOUNT returns the number of Bluetooth devices found.

4.3.3.26 _BTRADIOCOUNT

_BTRADIOCOUNT returns the number of Bluetooth radios installed on the system.

4.3.3.27 _BTSERVICECOUNT

_BTSERVICECOUNT returns the number of Bluetooth services present.

4.3.3.28 _CAPSLOCK

_CAPSLOCK returns the current state of the *Caps Lock* key on the keyboard:

value	toggled status
1	ON
0	OFF

4.3.3.29 _CDROMS

_CDROMS returns a space-delimited list of the CD-ROM drives on the system.

4.3.3.30 _CHILDPID

_CHILDPID returns the process ID of the most recent child process.

4.3.3.31 _CI

_CI returns the insert mode cursor shape, as a percentage (**0** to **100**).

See also SETDOS /S and the Insert Cursor configuration option.

4.3.3.32 _CMDLINE

_CMDLINE returns the current command line. (This is most useful in key aliases.) If you specify it on the command line, it is expanded to the contents of the command line (not including the %_cmdline variable itself).

Example:

echo one two three %_cmdline

will return:

one two three echo one two three

4.3.3.33 _CMDPROC

_CMDPROC returns the name of the current command processor (*TCC* or *TCC-RT*).

4.3.3.34 _CMDSPEC

_CMDSPEC returns the full pathname of the command processor.

4.3.3.35 _CO

_CO returns the overstrike mode cursor shape, as a percentage (**0** to **100**).

See also [SETDOS /S](#) and the Overstrike Cursor configuration option.

4.3.3.36 _CODEPAGE

_CODEPAGE returns the input code page used by the TCC-RT console.

See also [CHCP](#).

4.3.3.37 _COLUMN

_COLUMN is the current cursor column. The leftmost column is numbered **0**.

See also [_COLUMNS](#), [_ROW](#), and [_ROWS](#).

4.3.3.38 _COLUMNS

_COLUMNS returns the current number of virtual screen columns (for example, **80**).

See also [_COLUMN](#), [_ROW](#), and [_ROWS](#).

4.3.3.39 _CONSOLEB

_CONSOLEB returns the handle to the active console screen buffer.

See also @CONSOLEB.

4.3.3.40 _CONSOLEPIDS

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

4.3.3.41 _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.

4.3.3.42 _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.

_CPU is only supported in the 32-bit version of TCC-RT, and is obsolete. To determine the CPU type, revision, stepping level, and other details, use the @WININFO or @WMI functions.

4.3.3.43 _CPUUSAGE

_CPUUSAGE returns the current CPU usage, as a percent (**0** to **100**) .

4.3.3.44 _CTRL

_CTRL returns the status of the *Ctrl* keys:

value	status of selected key
1	at least one Ctrl key is depressed
0	neither is depressed

4.3.3.45 _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, _CWP, _CWPS, @CWD, and @CWDS.

4.3.3.46 _CWDS

_CWDS returns the current working directory in the format *d:\pathname*.

See also _CWD, _CWP, _CWPS, @CWD, and @CWDS.

4.3.3.47 _CWP

_CWP returns the current working directory in the format *\pathname* (without the drive letter).

See also _CWD, _CWDS, _CWPS, @CWD, and @CWDS.

4.3.3.48 _CWPS

_CWPS returns the current working directory in the format *\pathname* (without the drive letter).

See also _CWD, _CWDS, _CWP, @CWD, and @CWDS.

4.3.3.49 _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.

4.3.3.50 _DATETIME

_DATETIME returns the current date and time in the format yyyyMMddhhmmss. The date part is the same as _isodate without separators.

For the current UTC time, see _UTCDATETIME.

4.3.3.51 _DAY

_DAY returns the current day of the month (1 to 31).

4.3.3.52 _DETACHPID

_DETACHPID returns the process ID of the most recent process launched by the DETACH command.

4.3.3.53 _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 share name.

4.3.3.54 _DNAME

_DNAME returns the name of the file used to store file descriptions. It can be changed with the SETDOS /D command.

4.3.3.55 _DOS

_DOS returns the operating system type. *Take Command* returns a different value depending on the operating system, as follows:

Platform	*Take Command*
Windows Vista	`WINVISTA`
Windows 2008 Server	`WIN2008`
Windows 7	`WINDOWS7`
Windows 8	`WINDOWS8`
Windows 2012 Server	`WIN2012`
Windows 8.1	`WINDOWS81`
Windows 2012R2 Server	`WIN2012R2`
Windows 10	`WINDOWS10`
Windows 2016 Server	`WIN2016`

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 variable.

4.3.3.56 _DOSVER

_DOSVER returns the current operating system version.

4.3.3.57 _DOW

_DOW returns the first three characters of the name of the current day of the week (**Mon**, **Tue**, **Wed**, etc.).

_DOW returns the English name for the day of the week. For a localized version, see _IDOW.

4.3.3.58 _DOWF

_DOWF returns the full name of the day of the week for the current date (**Monday**, **Tuesday**, etc.).

_DOWF returns the English name for the day of the week. For a localized version, see _IDOWF.

4.3.3.59 _DOWI

_DOWI returns the current day of the week as an integer (**1** = Sunday, **2** = Monday, etc.).

4.3.3.60 _DOY

_DOY returns the current day of the year (1 to 366).

4.3.3.61 _DRIVES

_DRIVES returns a space-delimited list of the existing drives in the format:

    A: C: D: E:

_DRIVES only checks to see if the drive exists, not whether it is ready.

4.3.3.62 _DST

_DST returns 1 if daylight savings time is in effect, or 0 if it is not.

4.3.3.63 _DVDS

_DVDS returns a space-delimited list of the DVD drives on the system.

4.3.3.64 _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 command and the Batch Echo configuration option). The value returned by the _ECHO variable reflects the state applicable at the time the variable is queried.

4.3.3.65 _EDITMODE

_EDITMODE returns 0 if the line editor is in overstrike mode, or 1 if it is in insert mode.

4.3.3.66 _ELEVATED

_ELEVATED returns 1 if the TCC-RT process is elevated.

4.3.3.67 _EXECARRAY

_EXECARRAY returns the number of array elements assigned by the last @EXECARRAY function.

4.3.3.68 _EXECSTR

_EXECSTR returns the integer return code of the last @EXECSTR function.

4.3.3.69 _EXIT

_EXIT returns the reason for exiting TCC-RT. The possible values are:

0	EXIT command
2	CLOSE_EVENT
5	LOGOFF_EVENT
6	SHUTDOWN_EVENT

4.3.3.70 _EXPANSION

_EXPANSION returns the current expansion mode (i.e., SETDOS /X). 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**.

Note that %_expansion will return **%_expansion** if you turn off variable expansion (setdos /x-3)!

4.3.3.71 _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 for details.

4.3.3.72 _FILEARRAY

_FILEARRAY returns the number of array elements assigned by the last @FILEARRAY function.

4.3.3.73 _FTPERROR

_FTPERROR returns the error code of the last error reported by FTP. 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)

4.3.3.74 _GPSALT

_GPSALT returns the altitude (from sea level) in meters.

The GPS internal variables require a GPS (internal or external) that supports the Windows Sensor Location APIs. Note that your GPS may not support all of the variables. If a value is not supported Windows will return "Element not found". Unless otherwise specified, the values returned are decimal numbers (for example, 38.735122).

4.3.3.75 _GPSAZIMUTH

_GPSAZIMUTH returns a space-delimited list of the the azimuth of each satellite in view.

The GPS internal variables require a GPS (internal or external) that supports the Windows Sensor Location APIs. Note that your GPS may not support all of the variables. If a value is not supported Windows will return "Element not found". Unless otherwise specified, the values returned are decimal numbers (for example, 38.735122).

4.3.3.76 _GPSELEVATION

_GPSELEVATION returns a space-delimited list of the elevation of each satellite in view.

The GPS internal variables require a GPS (internal or external) that supports the Windows Sensor Location APIs. Note that your GPS may not support all of the variables. If a value is not supported Windows will return "Element not found". Unless otherwise specified, the values returned are decimal numbers (for example, 38.735122).

4.3.3.77 _GPSERRORRADIUS

_GPSERRORRADIUS returns the accuracy of the latitude and longitude values, in meters.

The GPS internal variables require a GPS (internal or external) that supports the Windows Sensor Location APIs. Note that your GPS may not support all of the variables. If a value is not supported Windows will return "Element not found". Unless otherwise specified, the values returned are decimal numbers (for example, 38.735122).

4.3.3.78 _GPSFIXQUALITY

_GPSFIXQUALITY returns the quality of the fix as an integer.

The GPS internal variables require a GPS (internal or external) that supports the Windows Sensor Location APIs. Note that your GPS may not support all of the variables. If a value is not supported Windows will return "Element not found". Unless otherwise specified, the values returned are decimal numbers (for example, 38.735122).

4.3.3.79 _GPSFIXTYPE

_GPSFIXTYPE returns the type of the fix as an integer.

The GPS internal variables require a GPS (internal or external) that supports the Windows Sensor Location APIs. Note that your GPS may not support all of the variables. If a value is not supported Windows will return "Element not found". Unless otherwise specified, the values returned are decimal numbers (for example, 38.735122).

4.3.3.80 _GPSHDOP

_GPSHDOP returns the horizontal dilution of precision.

The GPS internal variables require a GPS (internal or external) that supports the Windows Sensor Location APIs. Note that your GPS may not support all of the variables. If a value is not supported Windows will return "Element not found". Unless otherwise specified, the values returned are decimal numbers (for example, 38.735122).

4.3.3.81 _GPSHEADING

_GPSHEADING returns the true heading. See also _GPSMAGHEADING.

The GPS internal variables require a GPS (internal or external) that supports the Windows Sensor Location APIs. Note that your GPS may not support all of the variables. If a value is not supported Windows will return "Element not found". Unless otherwise specified, the values returned are decimal numbers (for example, 38.735122).

4.3.3.82 _GPSIDS

_GPSIDS returns a space-delimited list of IDs (integers) of the satellites in view.

The GPS internal variables require a GPS (internal or external) that supports the Windows Sensor Location APIs. Note that your GPS may not support all of the variables. If a value is not supported Windows will return "Element not found". Unless otherwise specified, the values returned are decimal numbers (for example, 38.735122).

4.3.3.83 _GPSLAT

_GPSLAT returns the latitude.

The GPS internal variables require a GPS (internal or external) that supports the Windows Sensor Location APIs. Note that your GPS may not support all of the variables. If a value is not supported Windows will return "Element not found". Unless otherwise specified, the values returned are decimal numbers (for example, 38.735122).

4.3.3.84 _GPSLON

_GPSLON returns the longitude.

The GPS internal variables require a GPS (internal or external) that supports the Windows Sensor Location APIs. Note that your GPS may not support all of the variables. If a value is not supported Windows will return "Element not found". Unless otherwise specified, the values returned are decimal numbers (for example, 38.735122).

4.3.3.85 _GPSMAGHEADING

_GPSHEADING returns the magnetic heading. See also _GPSHEADING.

The GPS internal variables require a GPS (internal or external) that supports the Windows Sensor Location APIs. Note that your GPS may not support all of the variables. If a value is not supported Windows will return "Element not found". Unless otherwise specified, the values returned are decimal numbers (for example, 38.735122).

4.3.3.86 _GPSNMEA

_GPSNMEA returns the NMEA sentence as a string..

The GPS internal variables require a GPS (internal or external) that supports the Windows Sensor Location APIs. Note that your GPS may not support all of the variables. If a value is not supported Windows will return "Element not found". Unless otherwise specified, the values returned are decimal numbers (for example, 38.735122).

4.3.3.87 _GPSOPMODE

_GPSOPMODE returns the GPS operation mode as an integer.

The GPS internal variables require a GPS (internal or external) that supports the Windows Sensor Location APIs. Note that your GPS may not support all of the variables. If a value is not supported Windows will return "Element not found". Unless otherwise specified, the values returned are decimal numbers (for example, 38.735122).

4.3.3.88 _GPSPDOP

_GPSPDOP returns the position dilution of precision..

The GPS internal variables require a GPS (internal or external) that supports the Windows Sensor Location APIs. Note that your GPS may not support all of the variables. If a value is not supported Windows will return "Element not found". Unless otherwise specified, the values returned are decimal numbers (for example, 38.735122).

4.3.3.89 _GPSPRNS

_GPSPRNS returns a space delimited list of the PRN numbers (integers) of the satellites in view.

The GPS internal variables require a GPS (internal or external) that supports the Windows Sensor Location APIs. Note that your GPS may not support all of the variables. If a value is not supported Windows will return "Element not found". Unless otherwise specified, the values returned are decimal numbers (for example, 38.735122).

4.3.3.90 _GPSSATSINVIEW

_GPSSATSINVIEW returns the number of satellites in view as an integer.

The GPS internal variables require a GPS (internal or external) that supports the Windows Sensor Location APIs. Note that your GPS may not support all of the variables. If a value is not supported Windows will return "Element not found". Unless otherwise specified, the values returned are decimal numbers (for example, 38.735122).

4.3.3.91 _GPSSATSUSED

_GPSSATSUSED returns the number of satellites used in the solution as an integer.

The GPS internal variables require a GPS (internal or external) that supports the Windows Sensor Location APIs. Note that your GPS may not support all of the variables. If a value is not supported Windows will return "Element not found". Unless otherwise specified, the values returned are decimal numbers (for example, 38.735122).

4.3.3.92 _GPSSELMODE

_GPSSELMODE returns the GPS selection mode as an integer.

The GPS internal variables require a GPS (internal or external) that supports the Windows Sensor Location APIs. Note that your GPS may not support all of the variables. If a value is not supported Windows will return "Element not found". Unless otherwise specified, the values returned are decimal numbers (for example, 38.735122).

4.3.3.93 _GPSSNR

_GPSSNR returns a space-delimited list of the signal to noise radio of each satellite in view.

The GPS internal variables require a GPS (internal or external) that supports the Windows Sensor Location APIs. Note that your GPS may not support all of the variables. If a value is not supported Windows will return "Element not found". Unless otherwise specified, the values returned are decimal numbers (for example, 38.735122).

4.3.3.94 _GPSSPEED

_GPSSPEED returns the speed in knots.

The GPS internal variables require a GPS (internal or external) that supports the Windows Sensor Location APIs. Note that your GPS may not support all of the variables. If a value is not supported Windows will return "Element not found". Unless otherwise specified, the values returned are decimal numbers (for example, 38.735122).

4.3.3.95 _GPSSTATUS

_GPSSTATUS returns the GPS status as an integer..

The GPS internal variables require a GPS (internal or external) that supports the Windows Sensor Location APIs. Note that your GPS may not support all of the variables. If a value is not supported Windows will return "Element not found". Unless otherwise specified, the values returned are decimal numbers (for example, 38.735122).

4.3.3.96 _GPSVDOP

_GPSVDOP returns the vertical dilution of precision.

The GPS internal variables require a GPS (internal or external) that supports the Windows Sensor Location APIs. Note that your GPS may not support all of the variables. If a value is not supported Windows will return "Element not found". Unless otherwise specified, the values returned are decimal numbers (for example, 38.735122).

4.3.3.97 _HDRIVES

_HDRIVES returns a space-delimited list of the hard (fixed) drives on the system.

4.3.3.98 _HLOGFILE

_HLOGFILE returns the name of the current history log file (or an empty string if LOG /H is OFF). See LOG for information on history logging.

4.3.3.99 _HOST

_HOST returns the host name for the local computer.

4.3.3.100 _HOUR

_HOUR returns the current hour (0 - 23) in local time.

For the current UTC time, see _UTCHOUR.

4.3.3.101 _HWPROFILE

_HWPROFILE returns the name of the current Windows hardware profile.

4.3.3.102 _HYPERV

_HYPERV returns 1 if TCC-RT is running inside a Hyper-V virtual machine.

4.3.3.103 _IDLETICKS

_IDLETICKS returns the number of milliseconds since the last user input.

4.3.3.104 _IDOW

_IDOW returns the 3-character abbreviation for the day of the week for the current date, in the current locale language.

See _DOW for the English language only version.

4.3.3.105 _IDOWF

_IDOWF returns the full name for the day of the week for the current date, in the current locale language.

For the English language only version, see _DOWF.

4.3.3.106 _IFTP

_IFTP returns **1** if an [IFTP](#) session is active, **0** if it is not.

4.3.3.107 _IFTPS

_IFTPS returns **1** if an SSL [IFTP](#) session is active, **0** if it is not.

4.3.3.108 _IMONTH

_IMONTH returns the abbreviated name for the current month, in the current locale language.

4.3.3.109 _IMONTHF

_IMONTHF returns the full name for the current month, in the current locale language.

4.3.3.110 _ININAME

_ININAME returns the fully qualified pathname of the INI file used by the current shell.

4.3.3.111 _INSERT

_INSERT returns 0 if the line editor is currently in overstrike mode, or 1 if it is in insert mode.

See also [_EDITMODE](#).

4.3.3.112 _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.

4.3.3.113 _IPADAPTER

_IPADAPTER returns the index of the current adapter.

4.3.3.114 _IPADAPTERS

_IPADAPTERS returns the number of adapters in the system.

4.3.3.115 _IPARPPROXY

_IPARPPROXY returns 1 if the local computer is acting as an ARP proxy, or 0 if it is not.

4.3.3.116 _IPDNS

_IPDNS returns 1 if DNS is enabled for the local computer, or 0 if it is not enabled.

4.3.3.117 _IPDNSOTHER

_IPDNSOTHER returns a space-delimited list of other DNS servers configured for the host machine. (The primary server is returned by %_IPDNSSERVER.)

4.3.3.118 _IPDNSSERVER

_IPDNSSERVER returns the default DNS server for the local computer.

4.3.3.119 _IPROUTING

_IPROUTING returns 1 if routing is enabled on the local computer, or 0 if it is not.

4.3.3.120 _IPV6

_IPV6 returns the IPv6 address of the local computer. If the computer has more than one NIC, **_IPv6** returns a space-delimited list of all IPv6 addresses.

Example:

```
echo %_ipv6
fe80::4cf7:928c:65b8:74ae fe80::68d5:b908:d058:3756
```

4.3.3.121 _ISFTP

_ISFTP returns **1** if an SSH IFTP session is active, **0** if it is not.

4.3.3.122 _ISODATE

_ISODATE returns the current local system date, in ISO 8601 format (**yyyy-mm-dd**）.

See also _DATE and _DATETIME.

4.3.3.123 _ISODOWI

_ISODOWI returns the ISO 8601 numeric day of the week (Monday=1, Sunday=7).

4.3.3.124 _ISOWDATE

_ISOWDATE returns the ISO 8601 current week date (yyyy-Www-d).

4.3.3.125 _ISOWEEK

_ISOWEEK returns the ISO 8601 week of year.

4.3.3.126 _ISOWYEAR

_ISOWYEAR returns the ISO 8601 week date year.

4.3.3.127 _KBHIT

_KBHIT returns **1** if one or more keystrokes are waiting in the keyboard buffer, or **0** if the keyboard buffer is empty.

4.3.3.128 _LALT

_LALT returns the status of the left *Alt* key on the keyboard:

value	key status
1	depressed
0	not depressed

See also _ALT and _RALT.

4.3.3.129 _LASTDIR

_LASTDIR returns the previous directory (from the directory history).

4.3.3.130 _LASTDISK

_LASTDISK returns the last valid drive letter (without a colon).

4.3.3.131 _LCTRL

_LCTRL returns the status of the Left Ctrl key on the keyboard:

value	key status
1	depressed
0	not depressed

See also _CTRL and _RCTRL.

4.3.3.132 _LOGFILE

_LOGFILE returns the name of the current command log file (or an empty string if LOG is OFF). See LOG for information on logging.

4.3.3.133 _LSHIFT

_LSHIFT returns the status of the left shift key on the keyboard:

value	key status
1	depressed
0	not depressed

See also _SHIFT and _RSHIFT.

4.3.3.134 _MINUTE

_MINUTE returns the current minute (0 - 59).

For the current UTC time, see _UTCMINUTE.

4.3.3.135 _MONITORS

_MONITORS returns the number of video displays.

4.3.3.136 _MONTH

_MONTH returns the current numeric month of the year (1 to 12).

4.3.3.137 _MONTHF

_MONTHF returns the full name of the current month (**January**, **February**, etc.).

4.3.3.138 _MSGBOX_CHECKBOX

_MSGBOX_CHECKBOX returns 1 if the user has checked the optional MSGBOX checkbox.

See also MSGBOX.

4.3.3.139 _NUMLOCK

_NUMLOCK reports the current state of the *Num Lock* key:

value	toggled status
1	**ON**
0	**OFF**

4.3.3.140 _OPENAFS

_OPENAFS returns **1** if the OpenAFS service is active, **0** if it is not.

See http://www.openafs.org for more information on OpenAFS.

4.3.3.141 _OSBUILD

_OSBUILD returns the Windows build number. The build number does not include the major or minor version.

4.3.3.142 _OSBUILDEX

_OSBUILDEX returns the Windows build and sub-build number. The build number does not include the major or minor version.

4.3.3.143 _PARENT

_PARENT returns the name of the parent process (the process that started TCC-RT).

4.3.3.144 _PBATCHNAME

_PBATCHNAME returns the name of the parent batch file. If there is no parent (the batch file was not CALL'd), it returns an empty string.

4.3.3.145 _PID

_PID returns the process ID number for the current TCC-RT process.

4.3.3.146 _PIPE

_PIPE returns **1** if the current process is running inside a pipe, and **0** otherwise.

4.3.3.147 _PPID

_PPID returns the process ID number of the parent process.

4.3.3.148 _RALT

_RALT returns the status of the right Alt key on the keyboard:

value	key status
1	depressed
0	not depressed

See also _ALT and _LALT.

4.3.3.149 _RCTRL

_RCTRL returns the status of the right Ctrl key on the keyboard:

value	key status
1	depressed
0	not depressed

See also _CTRL and _LCTRL.

4.3.3.150 _READY

_READY returns a space-delimited list of the currently ready (accessible) drives in the format :

C: D: E:

4.3.3.151 _REGISTERED

_REGISTERED returns the registered name of the user or an empty string if *Take Command* isn't registered.

4.3.3.152 _ROW

_ROW returns the current cursor row (for example, **0** for the top of the window).

4.3.3.153 _ROWS

_ROWS returns the current number of screen rows in the TCC-RT window (for example, **25**).

4.3.3.154 _RSHIFT

_RSHIFT returns the status of the right Shift key on the keyboard:

value	key status
1	depressed
0	not depressed

See also _SHIFT and _LSHIFT.

4.3.3.155 _RUBYTYPE

_RUBYTYPE returns the type of the Ruby VALUE returned by the last @RUBY call.

4.3.3.156 _RUBYVALUE

_RUBYVALUE returns the Ruby VALUE returned by the last @RUBY call.

4.3.3.157 _SCROLLLOCK

_SCROLLLOCK reports the current *scroll lock* state, which can be toggled using the *scroll lock* key:

value	toggled status
1	ON
0	OFF

4.3.3.158 _SECOND

_SECOND is the current second (0 - 59).

For the current UTC time, see _UTCSECOND.

4.3.3.159 _SELECTED

_SELECTED returns the first line of text highlighted in the *Take Command* tab window. If no text has been highlighted, **_SELECTED** returns an empty string.

4.3.3.160 _SERIALPORTS

_SERIALPORTS returns a space-delimited list of all of the available serial ports (COM1 - COMn). If there are no serial ports on the system, **_SERIALPORTS** returns an empty string.

4.3.3.161 _SERVICE

_SERVICE returns 1 if *TCC* was started as a service (TCC /N).

4.3.3.162 _SHELL

_SHELL is the current shell number. **_SHELL** will return 0 for a primary shell, or 1 (or higher) for a TCC-RT shell instance started by a parent *TCC-RT* process (either directly or via a pipe).

Note that the concept of shell numbers is now mostly obsolete in Windows.

4.3.3.163 _SHIFT

_SHIFT is the status the two *Shift* keys:

value	status of selected key
1	at least one is depressed
0	neither is depressed

4.3.3.164 _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.

4.3.3.165 _SHRALIAS

_SHRALIAS returns **1** if SHRALIAS is loaded, **0** if it is not.

4.3.3.166 _STARTPATH

_STARTPATH returns the startup directory for the current TCC-RT shell. (This is not necessarily the same as the location of the TCC-RT executable!)

4.3.3.167 _STARTPID

_STARTPID returns the process ID of the most recent process launched by the START command.

4.3.3.168 _STDIN

_STDIN returns **1** if STDIN points to the console, or **0** if it has been redirected.

4.3.3.169 _STDOUT

_STDOUT returns **1** if STDOUT points to the console, or **0** if it has been redirected.

4.3.3.170 _STDERR

_STDERR returns **1** if STDERR points to the console, or **0** if it has been redirected.

4.3.3.171 _STZN

_STZN returns the name of standard time in the current time zone.

See also [_STZO](#), [_TZN](#), and [_TZO](#).

4.3.3.172 _STZO

_STZO returns the offset in minutes from UTC for standard time in the current time zone.

See also [_STZN](#), [_TZN](#), and [_TZO](#).

4.3.3.173 _SYSERR

_SYSERR returns the error code of the last Windows system error.

4.3.3.174 _TCCINSTANCES

Returns the number of TCC-RT (version 21 or later) instances currently active.

4.3.3.175 _TCCRT

_TCCRT returns 1 if the current batch file is running in *TCC RT* (runtime), or 0 if it is running in *TCC*.

4.3.3.176 _TCCRUN

_TCCRUN returns the length of time the current TCC-RT session has been running (as a 64-bit integer, in 100ns increments).

4.3.3.177 _TCCSTART

_TCCSTART returns the time the current TCC-RT session was started (UTC, as a FILETIME, in 100ns increments).

4.3.3.178 _TCCVER

_TCCVER returns the current TCC-RT version (for example, 32).

See also: [_BUILD](#).

4.3.3.179 _TCEXIT

_TCEXIT returns the full pathname of the TCEXIT.* file, or an empty string if *TCC-RT* can't find TCEXIT.

Note that the string returned by _TCEXIT can change before TCEXIT is actually executed. (For example, if you modify the TCMD.INI settings.)

4.3.3.180 _TCFILTER

_TCFILTER returns the current filter in the *Take Command* File Explorer window if *TCC* is running in a tab window, or an empty string if it is not.

4.3.3.181 _TCFOLDER

_TCFOLDER returns the selected folder in the *Take Command* File Explorer window if *TCC* is running in a tab window, or an empty string if it is not.

4.3.3.182 _TCLISTVIEW

_TCLISTVIEW returns the selected entries in the **Take Command** File Explorer window if **TCC** is running in a tab window, or an empty string if it is not.

4.3.3.183 _TCMDINSTANCES

_TCMDINSTANCES returns the current number of **Take Command** (version 21 or later) instances.

4.3.3.184 _TCSTART

_TCSTART returns the full pathname of the TCSTART.* file, or an empty string if **TCC-RT** didn't find TCSTART.

4.3.3.185 _TCTAB

_TCTAB returns **1** if this **TCC** process is running in a **Take Command** tab window, or **0** if it is not.

4.3.3.186 _TCTABACTIVE

_TCTABACTIVE returns 1 if this **TCC** instance is the active tab in **Take Command**.

4.3.3.187 _TCTABS

_TCTABS returns the current number of **Take Command** tab windows (or 0 if **TCC** is not running in **Take Command**).

4.3.3.188 _TIME

_TIME returns the current system time in the format **hh:mm:ss**. The separator character may vary depending upon your country information.

4.3.3.189 _TRANSIENT

_TRANSIENT returns **1** if the current shell is transient (started with a **/C**, see Command Line Options for details), or **0** otherwise.

4.3.3.190 _TZN

_TZN returns the name of the current time zone.

See also _STZN, _STZO, and _TZO.

4.3.3.191 _TZO

_TZO returns the offset in minutes from UTC for the current time zone.

See also _STZN, _STZO, and _TZN.

4.3.3.192 _UNICODE

_UNICODE returns **1** if the shell is currently using Unicode for redirected output, **0** otherwise.

4.3.3.193 _USBS

_USBS : Returns a space-delimited list of the USB drives connected to the system.

4.3.3.194 _UTCDATE

_UTCDATE returns the current UTC date in the user's default format.

4.3.3.195 _UTCDATETIME

_UTCDATETIME returns the current UTC date and time.

For the local time, see _DATETIME.

4.3.3.196 _UTCHOUR

_UTCHOUR returns the current UTC hour.

For the local time, see _HOUR.

4.3.3.197 _UTCISODATE

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

For the current local date, see _ISODATE.

4.3.3.198 _UTCMINUTE

_UTCMINUTE returns the current UTC minute.

For the current local time, see _UTCMINUTE.

4.3.3.199 _UTCSECOND

_UTCSECOND returns the current UTC second.

For the current local time, see _SECOND.

4.3.3.200 _UTCTIME

_UTCTIME returns the current UTC time.

See _TIME to retrieve the current local time.

4.3.3.201 _VERMAJOR

_VERMAJOR returns the TCC-RT major version number (i.e., **10**).

4.3.3.202 _VERMINOR

_VERMINOR returns the TCC-RT minor version number (the tenths or hundredths digit).

For example, for **TCC-RT** *31.0,1* _VERMINOR will return **1.**

4.3.3.203 _VERSION

_VERSION returns the TCC-RT version in *major.minor* format (i.e., **31.01**).

4.3.3.204 _VIRTUALBOX

_VIRTUALBOX returns 1 if TCC-RT is running inside a VirtualBox virtual machine.

4.3.3.205 _VIRTUALPC

_VIRTUALPC returns 1 if TCC-RT is running inside a VirtualPC virtual machine. (Not supported in x64 TCC-RT.)

4.3.3.206 _VMWARE

_VMWARE returns 1 if TCC-RT is running inside a VMWare virtual machine. (Not supported in x64 TCC-RT.)

4.3.3.207 _VOLUME

_VOLUME returns the current volume level of the default audio device.

4.3.3.208 _VXPIXELS

_VXPIXELS returns the horizontal size of the virtual screen (including multiple monitors) in pixels.

4.3.3.209 _VYPIXELS

_VYPIXELS returns the vertical size of the virtual screen (including multiple monitors) in pixels.

4.3.3.210 _WINDIR

_WINDIR returns the pathname of the Windows directory.

4.3.3.211 _WINFGWINDOW

_WINFGWINDOW returns the title of the foreground window. (This may or may not be the Take Command or TCC-RT console window.)

4.3.3.212 _WINNAME

_WINNAME returns the computer name of the current system.

4.3.3.213 _WINSYSDIR

_WINSYSDIR returns the pathname of the Windows system directory.

4.3.3.214 _WINTICKS

_WINTICKS returns the number of milliseconds since Windows was started.

4.3.3.215 _WINTITLE

_WINTITLE returns the title of the current window.

4.3.3.216 _WINUSER

_WINUSER returns the name of the user currently logged on.

4.3.3.217 _WINVER

_WINVER returns the current Windows version number.

4.3.3.218 _WOW64

_WOW64 returns 1 if TCC-RT is running in the WOW64 environment (64-bit Windows). Note that this only applies to the old 32-bit versions of *TCC-RT*.

4.3.3.219 _WOW64DIR

_WOW64DIR returns the system Wow64 directory (x64 Windows only).

4.3.3.220 _X64

_X64 returns 1 if TCC-RT is the x64 (64-bit) version running on a 64-bit version of Windows.

4.3.3.221 _XEN

_XEN returns 1 if TCC-RT is running inside a Xen virtual machine.

4.3.3.222 _XMOUSE

_XMOUSE returns the column position of the most recent left mouse click. (Note that this will only work in a *Take Command* tab window, or if you have enabled the console mouse in a stand-alone TCC-RT session.)

4.3.3.223 _XPIXELS

_XPIXELS returns the number of horizontal pixels on the current physical display.

See also _YPIXELS.

4.3.3.224 _XWINDOW

_XWINDOW returns the width of the *Take Command* or TCC-RT window in pixels.

4.3.3.225 _YEAR

_YEAR returns the current year (1980 to 2099).

4.3.3.226 _YMOUSE

_YMOUSE returns the row position of the most recent left mouse click. (Note that this will only work in a *Take Command* tab window, or if you have enabled the console mouse in a stand-alone TCC-RT session.)

4.3.3.227 _YPIXELS

_YPIXELS returns the number of vertical pixels on the current physical display.

See also _XPIXELS.

4.3.3.228 _YWINDOW

_YWINDOW returns the height of the *Take Command* or TCC-RT window in pixels.

4.3.3.229 ERRORLEVEL

ERRORLEVEL is an alternate name (included for compatibility with CMD) for the ? 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 , some internal commands, e.g., **DIR**, also set this variable to the same value as the variable _?, 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 _? variable instead.

See also _?

4.3.4   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 and batch files to check on available system resources, manipulate strings and numbers, and work with files and filenames.

The variable functions built into TCC-RT are listed in alphabetical order in subsequent topics.

Note: The FUNCTION command can be used to create, edit, or display user-defined variable functions, and the UNFUNCTION to delete them.

For a list of Variable Functions organized by general categories of use, see Variable Functions by Category**.**

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 **[**.

Memory Size / Disk Space / File Size Units and Report Format

Some variable functions, such as @DISKFREE, 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
b	1		B	1		byte
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
p	1,000,000,000,000,000	10**15	P	1,125,899,906,842,624	2**50	petabyte
e	1,000,000,000,000,000,000	10**18	E	1,152,921,504,606,846,976	2**60	exabyte

You can include **commas** 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-RT is case sensitive.

Date Parameter Format

See the Date Formats 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 or @READY, the drive letter must be followed by a colon. The function will not work properly if you use the drive letter without the colon.

Functions Accessing File Handles

The @FILEREAD, @FILEWRITE, @FILEWRITEB, @FILESEEK, @FILESEEKL, and @FILECLOSE functions allow you to access files based on their file handle. These functions must be used only with file handles returned by @FILEOPEN, unless otherwise noted under the individual functions. If you use them with any other file handle you may damage files.

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 in commands.

Examples

You can use variable functions in a wide variety of ways depending on your needs. Here is an example to give you an idea of what's possible:

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[%$]`
```

4.3.4.1   Functions by Name

See also Functions by Category.

@ABS	Absolute value of number
@AFSCELL	OpenAFS cell name

@AFSMOUNT	OpenAFS mount point
@AFSPATH	Path in OpenAFS: 1, otherwise 0
@AFSSYMLINK	OpenAFS symbolic link
@AFSVOLID	OpenAFS volume ID
@AFSVOLNAME	OpenAFS volume name
@AGEDATE	Converts an age into date and time
@ALIAS	Value of an alias
@ALTNAME	Short name for the file
@ASCII	Set of ASCII-s for characters in string
@ARRAYINFO	Array variable information
@ASSOC	File association
@ATTRIB	Test or return file attributes
@AVERAGE	Average of a list of numbers

@B64DECODE	Decode Base64 file or string
@B64ENCODE	Encode file or string as Base64
@BALLOC	Allocate a binary buffer
@BFREE	Free a binary buffer
@BPEEK	Read a value from a binary buffer
@BPEEKSTR	Read a string from a binary buffer
@BPOKE	Write a value to a binary buffer
@BPOKESTR	Write a string to a binary buffer
@BREAD	Read from a file to a binary buffer
@BSIZE	Returns the size of a binary buffer
@BTDEVICEADDRESS	The Bluetooth address of the device
@BTDEVICEAUTHENTICATED	Returns 1 if the Bluetooth device has been authenticated
@BTDEVICECLASS	The device class for the Bluetooth device
@BTDEVICECONNECTED	Returns 1 if the Bluetooth device is connected
@BTDEVICELASTSEEN	The last time the Bluetooth device was seen
@BTDEVICELASTUSED	The last time the Bluetooth device was used
@BTDEVICENAME	The name of the Bluetooth device
@BTDEVICEREMEMBERED	Returns 1 if the Bluetooth radio is remembered
@BTRADIOADDRESS	The Bluetooth address of the radio
@BTRADIOCLASS	The device class of the Bluetooth radio
@BTRADIOCONNECTABLE	Returns 1 if the Bluetooth radio accepts connections
@BTRADIODISCOVERABLE	Returns 1 if the Bluetooth radio is discoverable
@BTRADIOMANUFACTURER	The manufacturer of the Bluetooth radio
@BTRADIONAME	The name of the Bluetooth radio
@BTRADIOSUBVERSION	The radio subversion of the Bluetooth radio
@BTSERVICEADDRESS	The Bluetooth address of the service
@BTSERVICECLASSID	The service class ID (UUID) of the Bluetooth service
@BTSERVICECOMMENT	A comment describing the Bluetooth service
@BTSERVICENAME	The name of the Bluetooth service
@BTSERVICEOTHERCLASSID	A list of other class IDs for the Bluetooth service
@BTSERVICEPORT	The port on which the Bluetooth service is running
@BTSERVICEPROTOCOL	The protocol used by the Bluetooth service
@BWRITE	Write from a binary buffer to a file

@CAPI	Call a _cdecl function in a DLL
@CAPS	Capitalize first character of each word
@CDROM	CD-ROM drive: 1, otherwise 0

@CEILING	Smallest integer not less than a number
@CHAR	Character string, given a set of ASCII-s
@CKSUM	Linux cksum compatible CRC32 checksum
@CLIP	Specified line from clipboard
@CLIPW	Write string to the clipboard
@CLIPWN	Write string to a TCC-RT clipboard CLIp0: - CLIP9:)
@COLOR	RGB value of a color
@COMMA	Insert commas into a number (thousands separators)
@COMPARE	Two files are identical: 1, otherwise 0
@CONSOLE	Identify console sessions
@CONSOLEB	Create or restore a console screen buffer
@CONVERT	Convert value from input base to output base
@COUNT	Number of times a character appears in a string
@CRC32	File CRC
@CWD	Current Working Directory of specified drive
@CWDS	Current Working Directory of specified drive, with trailing backslash

@DATE	Convert date to number of days
@DATECONV	Convert date to another format
@DATEFMT	Custom date formatting
@DAY	Day of month for date
@DEBUG	Write a string to the debugger
@DEC	Decrement a numeric value by 1
@DECIMAL	Decimal portion of a number
@DESCRIPT	File description
@DEVICE	Character device: 1, otherwise 0
@DIGITS	String is all digits: 1, otherwise 0
@DIRSTACK	Directory stack entry
@DISKFREE	Free disk space
@DISKTOTAL	Total disk space
@DISKUSED	Used disk space
@DOMAIN	Domain name of a computer
@DOW	Short name of day of week for date
@DOWF	Full name of day of week for date
@DOWI	Day of week number for date
@DOY	Day of year for date
@DRIVE	Drive name for filename
@DRIVETYPE	Type of a drive
@DRIVETYPEEX	Type of a drive

@EMAIL	Validate an email address
@ENUMSERVERS	Identify server names on a network
@ENUMSHARES	Identify sharenames on a server
@ERRTEXT	Windows error description
@EVAL	Arithmetic calculations
@EXEC	Execute a command and return its exit code
@EXECARRAY	Execute a command and store the results in an array variable
@EXECSTR	Execute a command and return the first output line
@EXETYPE	Application type
@EXPAND	All names that match filename
@EXT	File extension

@FIELD	Extract a field from a string
@FIELDS	Count fields in a string
@FILEAGE	File age (date and time)
@FILEARRAY	Read a file into an array
@FILECLOSE	Close a file handle
@FILEDATE	File date
@FILEHANDLE	Returns the filename for a handle
@FILELOCK	Returns PIDs with a lock on a file
@FILENAME	File name and extension
@FILEOPEN	Open a file handle
@FILEREAD	Read next line from a file
@FILEREADB	Read bytes from a file
@FILES	Number of files matching a wildcard
@FILESEEK	Move a file handle pointer to specified file position
@FILESEEKL	Move a file handle pointer to a specified line
@FILESIZE	Total size of files matching a wildcard
@FILETIME	File time
@FILETYPE	File encoding type (ASCII, UTF8, UTF16)
@FILEWRITE	Write next line to a file
@FILEWRITEB	Write data to a file
@FILTER	Removes non-matching characters from a string
@FINDCLOSE	Closes the search handle.
@FINDFIRST	Find first matching file
@FINDNEXT	Find next matching file
@FLOOR	Largest integer not larger than a number
@FOLDERS	Number of folders
@FORMAT	Formats data string according to format string
@FONT	Return console font information
@FORMATN	Format a numeric value
@FORMATNC	Format a numeric value and insert the thousands separator(s)
@FSTYPE	File system type (FAT, NTFS, CDFS, etc.)
@FTYPE	Open command string for file type
@FULL	Full file name with path
@FUNCTION	Definition of a function

@GETDATE	Select a date from a calendar
@GETDATETIME	Select a date and/or time from a date picker
@GETDIR	Prompt for a directory name.
@GETFILE	Prompt for a path and file name.
@GETFOLDER	Folder name from tree view.
@GROUP	User is member of group: 1, otherwise 0

@HEXDECODE	Decode hexadecimal file or string
@HEXENCODE	Encode file or string as hexadecimal
@HTMLDECODE	Decode an HTML escaped string
@HTMLENCODE	Encode a string for HTML

@IDOW	Short local name of day of week for date
@IDOWF	Full local name of day of week for date
@IF	Evaluates a conditional expression
@INC	Increment a numeric value by 1

@INDEX	Offset of string2 within string1
@INIREAD	Return an entry from an .INI file
@INIWRITE	Write an entry in an .INI file
@INSERT	Inserts string1 into string2
@INODE	File Inode (in hex)
@INSTR	Extract a substring
@INT	Integer part of a number
@IPADDRESS	Returns the numeric IP for a host name
@IPADDRESSN	Address of an adapter
@IPALIASES	Aliases of a host name
@IPBROADCAST	The broadcast address of an adapter
@IPDESC	Description of an adapter
@IPDHCP	DHCP server for an adapter
@IPDHCPENABLED	Returns 1 if the network adapter has DHCP enabled
@IPEXPIRES	Expiration date and time of the network adapter lease
@IPGATEWAY	Gateway for an adapter
@IPIPV6LL	IPv6 link local address of an adapter
@IPIPV6N	IPv6 address of an adapter
@IPNAME	Returns the host name for a numeric IP address
@IPNAMEN	Name of an adapter
@IPOBTAINED	Date and time of the network adapter lease
@IPOTHER	List of alternate addresses for the host
@IPOTHERL	List of other IP addresses leased by an adapter
@IPPHYSICAL	Physical address of an adapter
@IPPORT	Port number for a service
@IPSERVICEALIASES	Aliases for a service
@IPSTATUS	Current status of an adapter
@IPSUBNET	Subnet of an adapter
@IPTYPE	Type of an adapter
@IPWINS	Returns 1 if the adapter is using WINS
@IPWINSSERVER	Primary WINS server for an adapter
@IPWINSSERVER2	Secondary WINS server for an adapter
@IPZONEID	IPv6 Zone ID for an adapter
@ISALNUM	Test for alphanumeric characters
@ISALPHA	Test for alphabetic characters
@ISASCII	Test for ASCII characters
@ISCNTRL	Test for control characters
@ISDIGIT	Test for decimal digits
@ISFLOAT	Returns 1 if the string is a floating point number
@ISLOWER	Returns 1 if the string is only lower-case letters
@ISODOWI	ISO 8601 numeric day of week
@ISOWEEK	ISO 8601 numeric week of year
@ISOWYEAR	ISO 8601 numeric week date year
@ISPRIME	Test for prime number
@ISPRINT	Test for printable characters
@ISPROC	Returns 1 if the process is active; otherwise 0
@ISPUNCT	Test for punctuation characters
@ISUPPER	Returns 1 if the string is only upper-case letters
@ISSPACE	Test for white space characters
@ISXDIGIT	Test for hexadecimal digits

@JSONCLOSE	Close a JSON file
@JSONENDARRAY	Write the closing bracket of a JSON array
@JSONENDOBJECT	Write the closing brace of a JSON object
@JSONFLUSH	Flush the JSON parser buffers
@JSONHASXPATH	Returns 1 if the XPath exists in the JSON file, 0 otherwise
@JSONINPUT	Parse input string as JSON data
@JSONINSERTPROPERTY	Writes a value of a property at a selected position
@JSONINSERTVALUE	Insert a value at a selected position
@JSONNODENAMES	Returns the element names for the specified path
@JSONNODES	Returns the number of nodes for the specified path
@JSONOPEN	Open a JSON file
@JSONOUTPUT	Output JSON to string
@JSONPUTNAME	Write the name of a property
@JSONPUTPROPERTY	Writes a property and value
@JSONPUTRAW	Writes a raw JSON fragment
@JSONPUTVALUE	Write the value of a property
@JSONREMOVE	Remove the element or value set in XPath
@JSONRESET	Flush the JSON parser buffer & initialize parser to default state
@JSONSAVE	Save the modified JSON file to the output file
@JSONSETNAME	Set a new name for the element specified by XPath
@JSONSETVALUE	Set a new value for the element specified by XPath
@JSONSTARTARRAY	Write the opening bracket of a JSON array
@JSONSTARTOBJECT	Write the opening brace of a JSON object
@JSONXPATH	JSON XPath query
@JUNCTION	Directory referenced by the junction

@LABEL	Volume label
@LCS	Longest common sequence in two strings
@LEFT	Left end of string
@LEN	Length of a string
@LFN	Long name for a short filename
@LINE	Specified line from a file
@LINES	Count of lines in a file
@LINKS	Number of NTFS hard links for the file
@LOWER	Convert string to lower case
@LTRIM	Left trim specified characters.
@LUA	Execute a Lua expression

@MACADDRESS	MAC address of network interface
@MAKEAGE	Convert date and time to age
@MAKEDATE	Convert number of days to date
@MAKETIME	Convert number of seconds to time
@MAX	Largest integer in the list
@MD5	MD5 hash of a string or file
@MIN	Smallest integer in the list
@MONTH	Month for date
@MX	Email server for the specified user address

@NAME	File name without path or extension
@NUMERIC	Test if a string is numeric

@ODBCCLOSE	Close an ODBC session
@ODBCOPEN	Open an ODBC session
@ODBCQUERY	Query a SQL database
@OPTION	Current configuration option value
@OWNER	Return file owner

@PARSE	Parse the command line for switches
@PATH	File path without name
@PERL	Evaluate a Perl expression
@PID	PID for the specified process name
@PIDCOMMAND	Return startup command line for the specified process
@PIDUSER	Return user name for a PID
@PING	Response time from a host
@PINGR	Address of the host responding to the PING
@PLUGIN	Full pathname for plugin
@PLUGINVER	Plugin version number (major.minor.build)
@PPID	PID for the parent of the specified process
@PRIME	Generate a prime number
@PRIORITY	Priority class for the specified process
@PROCESSIO	Process I/O information
@PROCESSTIME	Process times (start, end, kernel mode, user mode)
@PSHELL	Evaluate a PowerShell expression
@PUNYDECODE	Decode a Punycode string or file
@PUNYENCODE	Encode a Punycode string or file
@PYTHON	Evaluate a Python expression

@QPDECODE	Decode using Quote-Printable MIME format
@QPENCODE	Encode using Quote-Printable MIME format
@QUOTE	Double quote the argument if necessary

@RANDOM	Generate a random integer
@READSCR	Read characters from the screen
@READY	Drive ready: 1, otherwise 0
@REGBREAD	Read a registry value to a binary buffer
@REGBWRITE	Write a registry value from a binary buffer
@REGCOPYKEY	Recursively copy a registry key to a new location
@REGCREATE	Create registry subkey
@REGDELKEY	Delete a registry key and its subkeys
@REGEX	Match a regular expression
@REGEXINDEX	Return the offset of a regular expression match
@REGEXIST	Test if a registry key exists
@REGEXSUB	Return nth matching regular expression group
@REGQUERY	Read value from registry
@REGSET	Write value to registry
@REGSETENV	Write value to registry and broadcast change.
@REGTYPE	Return type of registry variable
@REMOTE	Remote (network) drive: 1, otherwise 0
@REMOVABLE	Removable drive: 1, otherwise 0
@REPEAT	Repeat a character
@REPLACE	Replace string1 with string2 in text
@REREPLACE	Regular expression back reference substitution

@REVERSE	Reverse a string
@REXX	Value of executing an expression by REXX
@RIGHT	Right end of string.
@RTRIM	Removes specified trailing characters.
@RUBY	Evaluate a Ruby expression

@SCRIPT	Evaluate expression in an active scripting engine.
@SEARCH	Path search
@SELECT	Menu selection from file
@SELECTARRAY	Menu selection from array variable
@SERIAL	Serial number of a disk
@SERIALHW	Hardware serial number of a disk
@SERIALPORTCLOSE	Close the serial port
@SERIALPORTFLUSH	Flush the serial port buffer
@SERIALPORTOPEN	Open a serial port
@SERIALPORTREAD	Read the serial port buffer
@SERIALPORTWRITE	Write a string to the serial port buffer
@SERVER	Query server information
@SERVICE	Query service information
@SFN	Short name for a long filename
@SHA1	SHA1 checksum for the file
@SHA256	SHA2-256 checksum for the file
@SHA384	SHA2-384 checksum for the file
@SHA512	SHA2-512 checksum for the file
@SHFOLDER	Get Windows folder locations
@SIMILAR	Compare two strings for similarity
@SMCLOSE	Close a shared memory handle
@SMOPEN	Return a handle to shared memory
@SMPEEK	Read a value from shared memory
@SMPOKE	Write a value to shared memory
@SMREAD	Read a string from shared memory
@SMWRITE	Write a string to shared memory
@SNAPSHOT	Save a window or desktop as a BMP
@STRIP	Strips all characters in char from string
@SUBST	Substitute a string within another string
@SUBSTR	Extract a substring
@SUMMARY	Query or set the NTFS SummaryInformation stream
@SYMLINK	Target of a symbolic link
@SYSTEMTIME	System idle, kernel, or user time

@TALNUM	Number of alphanumeric characters in a string
@TALPHA	Number of alphabetic characters in a string
@TARCFILE	Compressed name of a file in a tar archive
@TARCOUNT	Number of files in a tar archive
@TARDFILE	Uncompressed name of a file in a tar archive
@TARFILEDATE	Date and time of a file in a tar archive
@TARFILESIZE	Size of a file in a tar archive
@TASCII	Number of 7-bit ASCII characters in a string
@TCL	Evaluate a Tcl expression
@TCNTRL	Number of ASCII control characters in a string
@TDIGIT	Number of digits (0-9) in a string

@TIME	Convert a time of day to number of seconds
@TIMER	Get split time from timer.
@TK	Evaluate a Tk (Tcl) script or expression
@TMP	Read a string from a TMP device
@TMPWN	Write a string to a TMP device
@TLOWER	Number of lower case characters in a string
@TPRINT	Number of printable characters in a string
@TPUNC	Number of punctuation characters in a string
@TRIM	Remove leading & trailing blanks from a string
@TRIMALL	Remove leading, trailing, and extra internal blanks from a string
@TRUENAME	Find true name of a file
@TRUNCATE	Truncate file at current position
@TSPACE	Number of white space characters in a string
@TUPPER	Number of upper case characters in a string
@TXDIGIT	Number of hexadecimal digits in a string

@UNC	UNC name of a file
@UNICODE	Numeric UNICODE value for a character
@UNIQUE	Create file with unique name
@UNQCLOSE	Close a UnQLite database
@UNQDELETE	Delete a key/value pair from a UnQLite database
@UNQKVB	Add a key/binary blob value pair to a UnQLite database
@UNQKVBA	Append a binary blob to the value of an existing UnQLite key/value pair
@UNQKVF	Add a key/file value pair to a UnQLite database
@UNQKVFA	Append a file to the value of an existing UnQLite key/value pair
@UNQKVS	Add a key/value pair to a UnQLite database
@UNQKVSA	Append a string to the value of an existing UnQLite key/value pair
@UNQOPEN	Open a UnQLite database
@UNQREADB	Read a binary value from a UnQLite database
@UNQREADF	Read a value from a UnQLite database and write it to a file
@UNQREADS	Read a string value from a UnQLite database
@UNQUOTE	Remove double quotes from a filename
@UNQUOTES	Remove leading and trailing double quotes
@UPPER	Convert string to upper case
@URLDECODE	Decode an URL string
@URLENCODE	Encode an URL string
@USB	If drive is USB : 1, else 0
@UTF8DECODE	Decode a UTF8 file or string
@UTF8ENCODE	Encode a file or string as UTF8
@UUDECODE	Decode a UU Encoded file
@UUENCODE	Encode a file as UU Encoded
@UUID	Returns a UUID (GUID)

@VARTYPE	Return the variable type
@VERINFO	Executable file version information
@VERSION	Returns a versioned replacement for a filename

@WATTRIB	Test or return file attributes
@WILD	Compares strings using wildcards
@WINAPI	Call a Windows API function
@WINCLASS	Title of first window with class name

@WINCLIENTSIZE	Client window size
@WINEXENAME	Executable name for window
@WININFO	Current system information
@WINMEMORY	Windows memory information
@WINMETRICS	Windows system metrics
@WINPATH	Convert WSL filename to Windows
@WINPID	Process ID for a window
@WINPOS	Window position
@WINSIZE	Window size
@WINSTATE	Current state of window
@WINSYSTEM	Set/get windows system parameters
@WMI	Query WMI
@WORD	Extract a word from a string
@WORDS	Count words in a string
@WORKGROUP	Workgroup name of a computer
@WSLPATH	Convert Windows filename to WSL

@XMLCLOSE	Close an XML file previously opened by @XMLOPEN
@XMLENDELEMENT	Write the closing tab of an XML element
@XMLFLUSH	Flush the XML parser buffers
@XMLGETATTR	Return the value of the specified attribute
@XMLHASXPATH	Return 1 if the XPath exists in the XML file, 0 otherwise
@XMLINPUT	Parse input string as XML data
@XMONODENAMES	Element names for the specified path
@XMLNODES	Return the number of nodes (children) for the specified path in an XML file
@XMLOPEN	Open an XML file for use by @XMLXPATH and/or @XMLNODES
@XMLOUTPUT	Output XML to a string
@XMLPUTATTR	Write an XML attribute
@XMLPUTCDATA	Write an XML CDATA block
@XMLPUTCOMMENT	Write an XML comment block
@XMLPUTELEMENT	Write a simple XML element with no attributes and a value
@XMLPUTSTRING	Write text inside an XML element
@XMLREMOVECHILDREN	Removes the children of the element at the specified path
@XMLREMOVEELEMENT	Removes the element and its children at the specified XPath
@XMLRESET	Flush the XML parser buffers and initialize parser to default state
@XMLSAVE	Save the modified XML document
@XMLSTARTELEMENT	Write the opening tab of a new XML element
@XMLXPATH	Return text of XML element

@YDECODE	Decode a Y Encoded file or string
@YEAR	Year for date
@YENCODE	Encode a file or string as Y Encoded

@ZIPCFILE	The compressed name of a file in a zip archive
@ZIPCOMMENT	The comment for a zip archive
@ZIPCOUNT	The number of files in a zip archive
@ZIPDFILE	The decompressed name of a file in a zip archive
@ZIPFILECOMMENT	The comment (description) of a file in a zip archive
@ZIPFILECRC	The CRC of a file in a zip archive
@ZIPFILEDATE	The date and time of a file in a zip archive
@ZIPCFILESIZE	The compressed size of a file in a zip archive

@ZIPDFILESIZE	The decompressed size of a file in a zip archive

4.3.4.2 Functions by Category

See also Functions by Name.

This list gives a one-line description of all built-in Variable Functions, and a cross reference which selects a separate help topic on that function where you will find the detailed syntax and description.

- Binary buffers
- Compression
- Dates and times
- Drives and devices
- File content
- File names
- File properties

- Input dialog boxes
- Network properties
- Numbers and arithmetic
- Strings and characters
- System status
- Utility

Note: many functions have functionality that covers several categories.

System status

@ASSOC	File association for the extension
@CLIP	Specified line from clipboard
@CLIPW	Write string to the clipboard
@CONSOLE	Identify console sessions
@CONSOLEB	Create or restore console screen buffers
@ERRTEXT	Windows error description
@FTYPE	Open command string for the file type
@ISPROC	Returns 1 if a process is active; otherwise 0
@PID	Process ID for the process name
@PIDCOMMAND	Startup command line for a process
@PIDUSER	User name for a PID
@PPID	Process ID of the parent of the specified process
@PRIORITY	Priority class for a process
@PROCESSIO	Process I/O information
@PROCESSTIME	Process times (start, end, kernel mode, user mode)
@READSCR	Read characters from the screen
@REGBREAD	Read registry value into a binary buffer
@REGBWRITE	Write registry value from a binary buffer
@REGCOPYKEY	Recursively copy a registry key to a new location
@REGCREATE	Create registry subkey
@REGDELKEY	Delete a registry key and its subkeys
@REGEXIST	Test if a registry key exists
@REGQUERY	Read value from registry
@REGSET	Write value to registry
@REGSETENV	Write value to registry and broadcast change.
@REGTYPE	Type of registry variable
@SERIALPORTCLOSE	Close the serial port
@SERIALPORTFLUSH	Flush the serial port buffer
@SERIALPORTOPEN	Open a serial port
@SERIALPORTREAD	Read the serial port buffer
@SERIALPORTWRITE	Write a string to the serial port buffer
@SYSTEMTIME	System times (idle, kernel, and user)

@WINCLASS	Title of first window with classname
@WINCLIENTSIZE	Client window size
@WINEXENAME	Executable name for window
@WININFO	Current system information
@WINMEMORY	Windows memory information
@WINMETRICS	Windows system metrics
@WINPID	Process ID for window
@WINPOS	Window position
@WINSIZE	Window size
@WINSTATE	Current state of window
@WINSYSTEM	Set/get windows system parameters
@WINTITLE	Window title of PID

Directories, drives and devices

@CDROM	CD-ROM drive: 1, otherwise 0
@CWD	Current Working Directory of specified drive
@CWDS	Current Working Directory of specified drive, with trailing backslash
@DEVICE	Character device: 1, otherwise 0
@DISKFREE	Free disk space
@DISKTOTAL	Total disk space
@DISKUSED	Used disk space
@DRIVE	Drive name for filename
@DRIVETYPE	Type of drive (hard drive, CD-ROM, etc.)
@DRIVETYPEEX	Type of drive (hard drive, CD-ROM, etc.)
@FSTYPE	File system type (FAT, NTFS, CDFS, etc.)
@JUNCTION	Directory referenced by the junction
@LABEL	Volume label
@READY	Drive ready: 1, otherwise 0
@REMOTE	Remote (network) drive: 1, otherwise 0
@REMOVABLE	Removable drive: 1, otherwise 0
@SERIAL	Serial number of a disk
@SERIALHW	Hardware serial number of a disk
@SHFOLDER	Windows folder locations
@SYMLINK	Target of a symbolic link
@USB	Drive is USB : 1, else 0

File content

@B64DECODE	Decode a Base64 file or string
@B64ENCODE	Encode a file or string as Base64
@CKSUM	Linux cksum-compatible CRC32
@COMPARE	Compare two files
@CRC32	File CRC
@FILEARRAY	Read a file into an array
@FILECLOSE	Close a file handle
@FILEOPEN	Open a file handle
@FILEREAD	Read next line from a file
@FILEREADB	Read bytes from a file
@FILESEEK	Move a file handle pointer
@FILESEEKL	Move a file handle pointer to a specified line
@FILETYPE	Returns the file encoding (ASCII, UTF8, UTF16)

@FILEWRITE	Write next line to a file
@FILEWRITEB	Write data to a file handle
@HEXDECODE	Decode a hexadecimal file or string
@HEXENCODE	Encode a file or string as hexadecimal
@INIREAD	Return an entry from an .INI file
@INIWRITE	Write an entry in an .INI file
@INODE	Inode value for a file
@LINE	Specified line from a file
@LINES	Count lines in a file
@LINKS	Number of NTFS hard links for a file
@MD5	MD5 hash of a string or file
@PUNYDECODE	Decode a Punycode string or file
@PUNYENCODE	Encode a Punycode string or file
@QPDECODE	Decode using Quote-Printable MIME format
@QPENCODE	Encode using Quote-Printable MIME format
@SHA1	SHA1 checksum for a file
@SHA256	SHA2-256 checksum for a file
@SHA384	SHA2-384 checksum for a file
@SHA512	SHA2-512 checksum for a file
@SUMMARY	NTFS SummaryInformation stream for a file
@TRUNCATE	Truncate file at current position
@UTF8DECODE	Decode a UTF8 file or string
@UTF8ENCODE	Encode a file or string as UTF8
@UUDECODE	Decode a UU Encoded file
@UUENCODE	Encode a file as UU Encoded
@VERINFO	Executable file version information
@YDECODE	Decode a Y Encoded file or string
@YENCODE	Encode a file or string as Y Encoded

File names

@ALTNAME	Short name for the file.
@EXPAND	All names that match filename
@EXT	File extension
@FILEHANDLE	Filename for a handle
@FILENAME	File name and extension
@FULL	Full file name with path
@LFN	Long name for a short filename
@NAME	File name without path or extension
@PATH	File path without name
@QUOTE	Double quote a filename
@SFN	Short name for a long filename
@SEARCH	Path search
@TRUENAME	True name of a file
@UNC	UNC name of a file
@UNIQUE	Create file with unique name
@UNQUOTE	Remove double quotes from a filename
@UNQUOTES	Remove leading and trailing double quotes
@VERSION	Returns a versioned filename

File properties

@ATTRIB	Test or return file attributes
@DESCRIPT	File description
@EXETYPE	Application type
@FILEAGE	File age (date and time)
@FILEDATE	File date
@FILELOCK	Return PIDs with a lock on the file
@FILES	Number of files matching a wildcard
@FILESIZE	Total size of files matching a wildcard
@FILETIME	File time
@FINDCLOSE	Closes the search handle.
@FINDFIRST	Find first matching file
@FINDNEXT	Find next matching file
@INODE	Inode value for a file
@LINKS	Number of NTFS hard links for a file
@OWNER	File owner
@SEARCH	Path search
@SUMMARY	NTFS SummaryInformation stream for a file
@TRUENAME	True name for a file
@UNIQUE	Create file with unique name
@VERINFO	Executable file version information
@WATTRIB	Test or return file attributes

Strings and characters

@ASCII	List of ASCII-s for characters in string
@CAPS	Capitalize first character of each word
@CHAR	Character string, given a set of ASCII-s
@COUNT	Counts occurrences of a character in a string
@EMAIL	Validate an email address
@EXECARRAY	Execute a command and store the results in an array variable
@EXECSTR	Execute a command and return the output line
@FIELD	Extract a field from a string
@FIELDS	Count fields in a string
@FILTER	Removes non-matching characters from a string
@FORMAT	Formats data string according to format string
@HTMLDECODE	Decode an HTML string
@HTMLENCODE	Encode a string for HTML
@INDEX	Offset of string2 within string1
@INSERT	Insert string1 into string2
@INSTR	Extract a substring
@ISALNUM	Test for alphanumeric characters
@ISALPHA	Test for alphabetic characters
@ISASCII	Test for ASCII characters
@ISCNTRL	Test for control characters
@ISDIGIT	Test for decimal digits
@ISFLOAT	Returns 1 if the string is a floating point number
@ISLOWER	Returns 1 if the string is all lower case
@ISPRINT	Test for printable characters
@ISPUNCT	Test for punctuation characters
@ISSPACE	Test for white space characters
@ISUPPER	Returns 1 if the string is all upper case
@ISXDIGIT	Test for hexadecimal digits

@LCS	Longest common sequence in two strings
@LEFT	Left end of string
@LEN	Length of a string
@LOWER	Convert string to lower case
@LTRIM	Trims specified leading characters.
@MD5	MD5 hash of a string or file
@MX	Email server for a user address
@PARSE	Parse the command line for switches
@QUOTE	Double quote a string
@REGEX	Return a Regular Expression test
@REGEXINDEX	Return the offset of a regular expression match
@REGEXSUB	Return the nth matching group of a regular expression test
@REPEAT	Repeat a character
@REPLACE	Replace string1 with string2 in text
@REREPLACE	Regular expression back reference substitution
@REVERSE	Reverse a string
@RIGHT	Right end of string
@RTRIM	Trims specified trailing characters.
@SIMILAR	Test similarity between two strings
@STRIP	Strips all characters in char from string
@SUBST	Substitute a string within another string
@SUBSTR	Older version of @INSTR to extract a substring
@TALNUM	Number of alphanumeric characters in a string
@TALPHA	Number of alphabetic characters in a string
@TASCII	Number of 7-bit ASCII characters in a string
@TCNTRL	Number of ASCII control characters in a string
@TDIGIT	Number of digits (0-9) in a string
@TLOWER	Number of lower case characters in a string
@TPRINT	Number of printable characters in a string
@TPUNCT	Number of punctuation characters in a string
@TRIM	Remove leading and trailing blanks from a string
@TRIMALL	Removing leading, trailing, and extra internal blanks from a string
@TSPACE	Number of white space characters in a string
@TUPPER	Number of upper case characters in a string
@TXDIGIT	Number of hexadecimal digits in a string
@UNICODE	List of UNICODEs for characters in string
@UNQUOTE	Remove double quotes from a string
@UNQUOTES	Remove leading and trailing double quotes
@UPPER	Convert string to upper case
@URLDECODE	Decode an URL string
@URLENCODE	Encode an URL string
@WILD	Compares strings using wildcards
@WORD	Extract a word from a string
@WORDS	Count words in a string

Binary buffers and shared memory

@BALLOC	Allocate a binary buffer
@BFREE	Free a binary buffer
@BPEEK	Read a value from a binary buffer
@BPEEKSTR	Read a string from a binary buffer
@BPOKE	Write a value to a binary buffer

@BPOKESTR	Write a string to a binary buffer
@BREAD	Read from a file to a binary buffer
@BSIZE	Returns the size of a binary buffer
@BWRITE	Write from a binary buffer to a file
@SMCLOSE	Close a handle to shared memory
@SMOPEN	Open a handle to shared memory
@SMPEEK	Read a value from shared memory
@SMPOKE	Write a value to shared memory
@SMREAD	Read a string from shared memory
@SMWRITE	Write a string to shared memory

Numbers and arithmetic

@ABS	Absolute value of n
@AVERAGE	Average of a list
@CEILING	Smallest integer not less than n
@COMMA	Insert commas (thousands separators) into a numeric string
@CONVERT	Convert value from input base to output base
@DEC	Decrement a numeric value by 1
@DECIMAL	Decimal fraction portion of a number
@DIGITS	Tests if string is all digits
@EVAL	Arithmetic calculations
@FORMATN	Format a numeric value
@FORMATNC	Format a numeric value and insert thousands separators
@FLOOR	Largest integer not larger than n
@INC	Increment a numeric value by 1
@INT	Integer part of a number
@ISPRIME	Test if a number is a prime
@MAX	Largest integer in the list
@MIN	Smallest integer in the list
@NUMERIC	Test if a string is numeric
@PRIME	Generate a prime number
@RANDOM	Generate a random integer

Dates and times

@AGEDATE	Converts an age into date and time
@DAY	Day of month for date
@DATE	Convert date to number of days
@DATECONV	Convert date formats
@DATEFMT	Custom date/time formatting
@DOW	Short name of day of week for date
@DOWF	Full name of day of week
@DOWI	Day of week as integer
@DOY	Day of year for date
@GETDATE	Select a date from a calendar
@GETDATETIME	Select a date and/or time from a date picker
@IDOW	Short localized name of day of week for date
@IDOWF	Full localized name of day of week for date
@ISODOWI	ISO 8601 numeric day of week
@ISOWEEK	ISO 8601 numeric week of year
@ISOWYEAR	ISO 8601 numeric week date year

@MAKEAGE	Convert date and time to age
@MAKEDATE	Convert number of days to date
@MAKETIME	Convert number of seconds to time
@MONTH	Month in specified date
@TIME	Convert time to number of seconds
@YEAR	Year for date

Input dialog boxes

@GETDIR	Prompt for a directory name.
@GETFILE	Prompt for a path and file name.
@GETFOLDER	Folder name from tree view.
@SELECT	Menu selection

Network properties

@AFSCELL	OpenAFS cell name for a path
@AFSMOUNT	OpenAFS mount point for a path
@AFSPATH	Path is in OpenAFS: 1, otherwise 0
@AFSSYMLINK	OpenAFS symbolic link for a path
@AFSVOLID	OpenAFS volume ID for a path
@AFSVOLNAME	OpenAFS volume name for a path
@BTDEVICEADDRESS	The Bluetooth address of the device
@BTDEVICEAUTHENTICATED	Returns 1 if the Bluetooth device is authenticated
@BTDEVICECLASS	The device class for the Bluetooth device
@BTDEVICECONNECTED	Returns 1 if the Bluetooth device is connected
@BTDEVICELASTSEEN	The last time the Bluetooth device was seen
@BTDEVICELASTUSED	The last time the Bluetooth device was used
@BTDEVICENAME	The name of the Bluetooth device
@BTDEVICEREMEMBERED	Returns 1 if the Bluetooth radio is remembered
@BTRADIOADDRESS	The Bluetooth address of the radio
@BTRADIOCLASS	The device class of the Bluetooth radio
@BTRADIOCONNECTABLE	Returns 1 if the Bluetooth radio accepts connections
@BTRADIODISCOVERABLE	Returns 1 if the Bluetooth radio is discoverable
@BTRADIOMANUFACTURER	The manufacturer of the Bluetooth radio
@BTRADIONAME	The name of the Bluetooth radio
@BTRADIOSUBVERSION	The radio subversion of the Bluetooth radio
@BTSERVICEADDRESS	The Bluetooth address of the service
@BTSERVICECLASSID	The service class ID (UUID) of the Bluetooth service
@BTSERVICECOMMENT	A comment describing the Bluetooth service
@BTSERVICENAME	The name of the Bluetooth service
@BTSERVICEOTHERCLASSID	A list of other class IDs for the Bluetooth service
@BTSERVICEPORT	The port on which the Bluetooth service is running
@BTSERVICEPROTOCOL	The protocol used by the Bluetooth service
@DOMAIN	Domain name of a computer
@ENUMSERVERS	Identify server names on a network
@ENUMSHARES	Identify sharenames on a server
@GROUP	User is member of group: 1, otherwise 0
@IPADDRESS	Returns the numeric IP for a host name
@IPADDRESSN	Address of an adapter
@IPALIASES	Aliases of a host name
@IPBROADCAST	The broadcast address of an adapter

@IPDESC	Description of an adapter
@IPDHCP	DHCP server for an adapter
@IPDHCPENABLED	Returns 1 if the network adapter has DHCP enabled
@IPEXPIRES	Expiration date and time of the network adapter lease
@IPGATEWAY	Gateway for an adapter
@IPIPV6LL	IPv6 link local address of an adapter
@IPIPV6N	IPv6 address of an adapter
@IPNAME	Returns the host name for a numeric IP address
@IPNAMEN	Name of an adapter
@IPOBTAINED	Date and time of the network adapter lease
@IPOTHER	List of alternate addresses for the host
@IPOTHERL	List of other IP addresses leased by an adapter
@IPPHYSICAL	Physical address of an adapter
@IPPORT	Port number for a service
@IPSERVICEALIASES	Aliases for a service
@IPSTATUS	Current status of an adapter
@IPSUBNET	Subnet of an adapter
@IPTYPE	Type of an adapter
@IPWINS	Returns 1 if the adapter is using WINS
@IPWINSSERVER	Primary WINS server for an adapter
@IPWINSSERVER2	Secondary WINS server for an adapter
@IPZONEID	IPv6 Zone ID for an adapter
@PING	Response time from a host
@PINGR	Address of the host responding to the PING

JSON Parsing

@JSONCLOSE	Close a JSON file
@JSONENDARRAY	Write the closing bracket of a JSON array
@JSONENDOBJECT	Write the closing brace of a JSON object
@JSONFLUSH	Flush the JSON parser buffers
@JSONHASXPATH	Returns 1 if the XPath exists in the JSON file, 0 otherwise
@JSONINPUT	Parse input string as JSON data
@JSONINSERTPROPERTY	Writes a value of a property at a selected position
@JSONINSERTVALUE	Insert a value at a selected position
@JSONNODENAMES	Returns the element names for the specified path
@JSONNODES	Returns the number of nodes for the specified path
@JSONOPEN	Open a JSON file
@JSONOUTPUT	Output JSON to a string after processing
@JSONPUTNAME	Write the name of a property
@JSONPUTPROPERTY	Write a property and value
@JSONPUTRAW	Write a raw JSON fragment
@JSONPUTVALUE	Write the value of a property
@JSONREMOVE	Remove the element or value set in XPath
@JSONRESET	Flush the JSON parser buffer & initialize parser to default state
@JSONSAVE	Save the modified JSON file to the output file
@JSONSETNAME	Set a new name for the element specified by XPath
@JSONSETVALUE	Set a new value for the element specified by XPath
@JSONSTARTARRAY	Write the opening bracket of a JSON array
@JSONSTARTOBJECT	Write the opening brace of a JSON object
@JSONXPATH	JSON XPath query

XML Parsing

@XMLCLOSE	Close an XML file previously opened by @XMLOPEN
@XMLENDELEMENT	Write the closing tab of an XML element
@XMLFLUSH	Flush the XML parser buffers
@XMLGETATTR	Return the value of the specified attribute
@XMLHASXPATH	Return 1 if the XPath exists in the XML file, 0 otherwise
@XMLINPUT	Parse input string as XML data
@XMONODENAMES	Element names for the specified path
@XMLNODES	Return the number of nodes (children) for the specified path in an XML file
@XMLOPEN	Open an XML file for use by @XMLXPATH and/or @XMLNODES
@XMLOUTPUT	Output XML to a string after processing
@XMLPUTATTR	Write an XML attribute
@XMLPUTCDATA	Write an XML CDATA block
@XMLPUTCOMMENT	Write an XML comment block
@XMLPUTELEMENT	Write a simple XML element with no attributes and a value
@XMLPUTSTRING	Write text inside an XML element
@XMLREMOVECHILDREN	Removes the children of the element at the specified path
@XMLREMOVEELEMENT	Removes the element and its children at the specified XPath
@XMLRESET	Flush the XML parser buffers and initialize parser to default state
@XMLSAVE	Save the modified XML document
@XMLSTARTELEMENT	Write the opening tab of a new XML element
@XMLXPATH	Return text of XML element

Utility

@ALIAS	Value of an alias
@ARRAYINFO	Array variable information
@CAPI	Call a _cdecl function in a DLL
@CLIP	Specified line from clipboard
@CLIPW	Write string to the clipboard
@CLIPWN	Write string to a TCC-RT clipboard (CLIP0: - CLIP9:)
@COLOR	RGB value of a color
@DEBUG	Write a string to the debugger
@DIRSTACK	Display directory stack entry
@ERRTEXT	Windows error description
@EXEC	Execute a command, returns its exit code
@EXECSTR	Execute a command, returns its first output line
@FONT	Console font information
@FUNCTION	Definition of a function
@IF	Value dependent on a conditional expression
@LUA	Execute a Lua expression
@ODBCCLOSE	Close an ODBC session
@ODBCOPEN	Open an ODBC session
@ODBCQUERY	Query a SQL database
@OPTION	Current configuration option value
@PERL	Evaluate a Perl expression

@PLUGIN	Full pathname for plugin
@PLUGINVER	Plugin version number
@PSHELL	Execute a Powershell expression
@READSCR	Read characters from the screen
@REXX	Evaluate a REXX expression
@SCRIPT	Evaluate expression in active scripting engine
@SELECT	Menu selection from file
@SELECTARRAY	Menu selection from array variable
@SERVICE	Query service information
@SNAPSHOT	Save a window or the desktop to a BMP
@TCL	Execute a Tcl/tk command
@TIMER	Get split time from timer.
@TK	Execute a Tk script or expression
@TMP	Read a string from a TMP device
@TMPWN	Write a string to a TMP device
@UNQCLOSE	Close a UnQlite database
@UNQDELETE	Delete a key/value pair from a UnQlite database
@UNQKVB	Add a key/binary blob value pair to a UnQlite database
@UNQKVBA	Append a binary blob to the value of an existing UnQlite key/value pair
@UNQKVF	Add a key/file value pair to a UnQlite database
@UNQKVFA	Append a file to the value of an existing UnQlite key/value pair
@UNQKVS	Add a key/value pair to a UnQlite database
@UNQKVSA	Append a string to the value of an existing UnQlite key/value pair
@UNQOPEN	Open a UnQlite database
@UNQREADB	Read a binary value from a UnQlite database
@UNQREADF	Read a file value from a UnQlite database
@UNQREADS	Read a string value from a UnQlite database
@UUID	Create a UUID / GUID
@WINAPI	Call a Windows API function
@WMI	Query WMI

Compression and Decompression

@TARCOUNT	The number of files in a .tar archive
@TARCFILE	The compressed name of a file in a .tar archive
@TARDFILE	The decompressed name of a file in a .tar archive
@TARFILEDATE	The date and time of a file in a .tar archive
@TARFILESIZE	The (uncompressed) size of a file in a .tar archive
@ZIPCOUNT	The number of files in a .zip archive
@ZIPCOMMENT	The comment text for a .zip archive
@ZIPCFILE	The compressed name of a file in a .zip archive
@ZIPDFILE	The decompressed name of a file in a .zip archive
@ZIPFILECOMMENT	The comment (description) of a file in a .zip archive
@ZIPFILECRC	The CRC of a file in a .zip archive
@ZIPFILEDATE	The date and time of a file in a .zip archive
@ZIPCFILESIZE	The compressed size of a file in a .zip archive
@ZIPDFILESIZE	The decompressed size of a file in a .zip archive

4.3.4.3   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	*see below*	system default
1	mm/dd/yy	USA
2	dd/mm/yy	European
3	yy/mm/dd	Japanese
4	yyyy-mm-dd	ISO 8601
5	yyyy-Www-d	ISO 8601
6	yyyy-ddd	ISO 8601

Field Order

For codes **1...6** the field order is as shown above. For code **0** the field order will also be one of those shown above. TCC-RT 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 codes **4**, **5,** and **6**. 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

Codes **4**, **5,** and **6** (ISO 8601) uses a hyphen as the separator character. For the other formats, the default Windows date separator is returned.

4.3.4.4   @ABS

@ABS[*n***]** : Returns the absolute value of the number ***n.***

Examples:

```
echo %@abs[-1]
1

echo %@abs[123]
123
```

4.3.4.5   @AFSCELL

@AFSCELL[*path***]** : Returns the OpenAFS cell name for the path.

See http://www.openafs.org for more information on OpenAFS.

4.3.4.6 @AFSMOUNT

@AFSMOUNT[*path***]** : Returns the OpenAFS mount point for the pathname.

See http://www.openafs.org for more information on OpenAFS.

4.3.4.7 @AFSPATH

@AFSPATH[*path***]** : Returns 1 if the path is in the OpenAFS file system.

See http://www.openafs.org for more information on OpenAFS.

4.3.4.8 @AFSSYMLINK

@AFSSYMLINK[*path***]** : Returns the OpenAFS symbolic link for the path.

See http://www.openafs.org for more information on OpenAFS.

4.3.4.9 @AFSVOLID

@AFSVOLID[*path***]** : Returns the OpenAFS volume ID for the path.

See http://www.openafs.org for more information on OpenAFS.

4.3.4.10 @AFSVOLNAME

@AFSVOLNAME[*path***]** : Returns the OpenAFS volume name for the path.

See http://www.openafs.org for more information on OpenAFS.

4.3.4.11 @AGEDATE

@AGEDATE[*n[,d]***]** : Converts an age ***n*** into a date and time pair, formatted according to the current country settings, or as explicitly specified by ***d*** (see Date Display Formats). 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 (1,1,6) echo %n %@agedate[128551146920835000,%n]

1 05-13-08,01:11:32.083
2 13-05-08,01:11:32.083
3 08-05-13,01:11:32.083
4 2008-05-13,01:11:32.083
5 2008-W20-2,01:11:32.083
6 2008-134,01:11:32.083
```

See also: Time Stamps, @FILEAGE and @MAKEAGE.

4.3.4.12 @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 /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]
d:\path\myprog.exe –options
```

4.3.4.13 @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 of the full path.

Examples:

```
echo %@altname["Long Name.exe"]
LONGNA~1.EXE

echo %@altname["C:\Program Files\Microsoft Office"]
C:\PROGRA~1\MICROS~4

echo %@altname["%CommonProgramFiles"]
C:\PROGRA~1\COMMON~1
```

4.3.4.14 @ARRAYINFO

@ARRAYINFO[_arrayname,option_] : Returns information about the specified array.

arrayname - name of the array (defined by SETARRAY) to query

option - the type of information:

    0 - total number of dimensions
    1 - number of elements in the first dimension
    2 - number of elements in the second dimension
    3 - number of elements in the third dimension
    4 - number of elements in the fourth dimension
    5 - total number of elements

@ARRAYINFO will return -1 if the array doesn't exist.

Examples:

```
setarray array[5,10]
echo %@arrayinfo[array,0]
2

echo %@arrayinfo[array,2]
10
```

4.3.4.15 @ASCII

@ASCII[_string_**]** : Returns the space separated list of ASCII values of the characters in _string_. You can use the escape character (^) before a special character, e.g., a quote or greater than (>) sign, to include it in _string_.

Note: The @UNICODE 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.

4.3.4.16 @ASSOC

@ASSOC[_.ext[,u]_**]** : Returns the file association for the specified extension. If the optional second argument _u_ is specified, @ASSOC will look in HKCU\SOFTWARE\CLASSES.

Example:

```
echo %@assoc[.doc]
Word.Document.8
```

4.3.4.17 @ATTRIB

@ATTRIB[_filename[,-rhsadecijlopt[,p]]_**]** : If you do not specify any attributes, @ATTRIB returns the attributes of the specified file in the format **RHSADECIJNOFTVPU**, rather than **0** or **1**. If two or more parameters are specified, @ATTRIB returns a **1** if the specified file has all the matching attribute(s); otherwise it returns a **0**. If the optional third argument `,p` is included (partial match), then @ATTRIB will return **1** if any of the attributes match

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
- **F**   Sparse file
- **I**   Not content-indexed
- **J**   Junction or symbolic link
- **L**   Junction or symbolic link
- **N**   Normal

O   Offline
P   Pinned
T   Temporary
U   Unpinned
V   Virtualized

The extended attributes are displayed when @ATTRIB is invoked with a single parameter, but they cannot be specified when querying files (two or more parameters). To query files based on the extended attributes, see @WATTRIB.

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]). 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: @WATTRIB, Attributes Switches and the ATTRIB command.

Examples:

```
echo %@attrib["C:\Program Files\My Program\myfile.exe",rhs,p]

echo Attributes for myfile.exe: %@attrib[myfile.exe]
```

4.3.4.18 @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 (or @INT).

Example:

```
echo %@average[1 3 6 8 10 13 15]
8.0
```

4.3.4.19 @B64DECODE

@B64DECODE[s,*string***]** : Decode a Base64 string (MIME encoding format). Returns the decoded string.
@B64DECODE[*inputfile,outputfile***]** : Decode a Base64 file (MIME encoding format). Returns 0 if the output file was successfully written.

See also: @B64ENCODE

Example:

```
echo %@b64decode[s,dGhpcyBpcyBhIHN0cmluZw==]
this is a string

echo %@b64decode[data.file.b64,date.file]
```

4.3.4.20 @B64ENCODE

@B64ENCODE[s,*string***]** : Encode a base 64 string (MIME encoding format). Returns the encoded string
@B64ENCODE[*inputfile,outputfile***]** : Encode a base 64 file (MIME encoding format). Returns 0 if the output file was successfully written.

Example:

```
echo %@b64encode[s,this is a string]
dGhpcyBpcyBhIHN0cmluZw==

echo %@b64encode[data.file,date.file.b64]
```

4.3.4.21 @BALLOC

@BALLOC[*size***]** : Allocate a buffer for binary operations. @BALLOC returns a handle to the buffer, which must be used for the subsequent binary functions. The only limit on the number and size of binary buffers is the amount of virtual memory available.

Example:

```
set handle=%@balloc[128]
echo %handle
5d4f280
```

4.3.4.22 @BFREE

@BFREE[*handle***]** : Free a binary buffer previously allocated by @BALLOC.

Example:

```
set handle=%@balloc[128]
echo %@bfree[%handle]
```

4.3.4.23 @BPEEK

@BPEEK[*handle,offset,size***]** : Read a value from a binary buffer.

handle - a binary handle from @BALLOC

offset - the byte offset in the buffer (decimal or hex)

size - the size of the value to read (in bytes):

> 1 - character
> 2 - short
> 4 - int
> 8 - int64

Example:

```
set handle=%@balloc[128]
set value=%@bpeek[%handle,0,4]
```

4.3.4.24 **@BPEEKSTR**

@BPEEKSTR[*handle,offset,type,length*] : Read a string from a binary buffer.

handle - a binary handle from @BALLOC

offset - the byte offset in the buffer (decimal or hex)

type - the string type:

> a - ASCII
> u - Unicode

length - the maximum number of characters to read (decimal or hex)

Example:

```
set handle=%@balloc[128]
set value=%@bpeekstr[%handle,0,a]
```

4.3.4.25 **@BPOKE**

@BPOKE[*handle,offset,size,value*] : Write a value to a binary buffer.

handle - a binary handle from **@BALLOC**

offset - the byte offset in the buffer (decimal or hex)

size - the size of the value (in bytes):

> 1 - character
> 2 - short
> 4 - int
> 8 - int64

value - the value to poke

@BPOKE returns 0 on success.

Example:

```
set handle=%@balloc[128]
set value=%@bpoke[%handle,0,4,1234]
```

4.3.4.26 **@BPOKESTR**

@BPOKESTR[*handle,offset,type,string*] : Write a string to a binary buffer.

handle - a binary handle from @BALLOC

offset - the byte offset in the buffer (decimal or hex)

type - the type of the string to write:

a - ASCII
u - Unicode

string - the string to poke

@BPOKESTR returns 0 on success.

Example:

```
set handle=%@balloc[128]
set value=%@bpokestr[%handle,0,a,string value]
```

4.3.4.27 @BREAD

@BREAD[*handle,offset,filehandle,fileoffset,length*] : Read from a file to a binary buffer.

handle - a binary handle from @BALLOC

offset - the byte offset in the buffer (decimal or hex)

filehandle - a file handle opened for reading (from @FILEOPEN)

fileoffset - the read offset (from the current file position) (decimal or hex)

length - number of bytes to read (decimal or hex)

@BREAD returns the number of bytes actually read.

Example:

```
set fhandle=%@fileopen[filename,r]
set bhandle=%@balloc[128]
set value=%@bread[%bhandle,0,%fhandle,0,32]
```

4.3.4.28 @BSIZE

@BSIZE[*handle*] : Returns the size of a binary buffer allocated with @BALLOC.

Example:

```
set handle=%@balloc[128]
echo @bsize[%handle]
128
```

4.3.4.29 @BTDEVICEADDRESS

@BTDEVICEADDRESS[n] : The Bluetooth address of the device whose index is n. (Indexes range from 0 to _btdevicecount.)

See also _BTDEVICECOUNT.

Example:

```
echo %@btdeviceaddress[0]
```

```
04:52:C7:25:47:13
```

4.3.4.30 @BTDEVICEAUTHENTICATED

@BTDEVICEAUTHENTICATED[n] : Returns 1 if the Bluetooth device whose index is *n* is authenticated; otherwise it returns 0. Indexes range from 0 to %_btdevicecount.

Example:

```
echo %@btdeviceauthenticated[0]
0
```

4.3.4.31 @BTDEVICECLASS

@BTDEVICECLASS[n] : Returns the device class for the Bluetooth device whose index is *n*. Indexes range from 0 to %_btdevicecount.

Example:

```
echo %@btdeviceclass[0]
```

4.3.4.32 @BTDEVICECONNECTED

@BTDEVICECONNECTED[n] : Returns 1 if the Bluetooth device whose index is *n* is connected; otherwise it returns 0. Indexes range from 0 to %_btdevicecount.

Example:

```
echo %@btdeviceconnected[0]
```

4.3.4.33 @BTDEVICELASTSEEN

@BTDEVICELASTSEEN[n] : Returns the last time the Bluetooth device whose index is *n* was seen. Indexes range from 0 to %_btdevicecount. The format of the date is "mm/dd/yyyy hh:mm:ss".

Example:

```
echo %@btdevicelastseen[0]
06/12/2021 15:22:40
```

4.3.4.34 @BTDEVICELASTUSED

@BTDEVICELASTUSED[n] : Returns the last time the Bluetooth device whose index is *n* was used. Indexes range from 0 to %_btdevicecount. The format of the date is "mm/dd/yyyy hh:mm:ss".

Example:

```
echo %@btdevicelastused[0]
06/12/2021 15:22:40
```

4.3.4.35 @BTDEVICENAME

@BTDEVICENAME[n] : Returns the name of the Bluetooth device whose index is *n*. Indexes range from 0 to %_btdevicecount.

Example:

```
echo %@btdevicename[0]
```

4.3.4.36 @BTDEVICEREMEMBERED

@BTDEVICEREMEMBERED[n] : Returns 1 if the Bluetooth device whose index is *n* is a remembered device; otherwise it returns 0. Indexes range from 0 to %_btdevicecount.

Example:

```
echo %@btdeviceremembered[0]
0
```

4.3.4.37 @BTRADIOADDRESS

@BTRADIOADDRESS[n] : Returns the address of the Bluetooth radio whose index is *n* is discoverable. (Indexes range from 0 to _btdevicecount.)

Example:

```
echo %@btradioaddress[0]
00:E0:4C:70:3A:03
```

4.3.4.38 @BTRADIOCLASS

@BTRADIOCLASS[n] : Returns the device class of the Bluetooth radio whose index is *n* is discoverable. (Indexes range from 0 to _btdevicecount.)

Example:

```
echo %@btradioclass[0]
2752772
```

4.3.4.39 @BTRADIOCONNECTABLE

@BTRADIOCONNECTABLE[n] : Returns 1 if the Bluetooth radio whose index is *n* accepts incoming connections; otherwise returns 0. (Indexes range from 0 to _btdevicecount.)

Example:

```
echo %@btradioconnectable[0]
1
```

4.3.4.40 @BTRADIODISCOVERABLE

@BTRADIODISCOVERABLE[n] : Returns 1 if the Bluetooth radio whose index is *n* is discoverable; otherwise returns 0. (Indexes range from 0 to _btdevicecount.)

Example:

```
echo %@btradiodiscoverable[0]
1
```

4.3.4.41 @BTRADIOMANUFACTURER

@BTRADIOMANUFACTURER[n] : Returns the manufacturer of the Bluetooth radio whose index is *n*. (Indexes range from 0 to _btdevicecount.)

Example:

```
echo %@btradiomanufacturer[0]
93
```

4.3.4.42 @BTRADIONAME

@BTRADIONAME[n] : Returns the name of the Bluetooth radio whose index is *n*. (Indexes range from 0 to _btdevicecount.)

Example:

```
echo %@btradioname[0]
MSI-PC
```

4.3.4.43 @BTRADIOSUBVERSION

@BTRADIOSUBVERSION[n] : Returns the subversion of the Bluetooth radio whose index is *n*. (Indexes range from 0 to _btdevicecount.)

Example:

```
echo %@btradiosubversion[0]
25805
```

4.3.4.44 @BTSERVICEADDRESS

@BTSERVICEADDRESS[n] : Returns the address of the Bluetooth service whose index is *n*. (Indexes range from 0 to _btdevicecount.)

Example:

```
echo %@btserviceaddress[0]
```

4.3.4.45 @BTSERVICECLASSID

@BTSERVICECLASSID[n] : Returns the service class ID (UUID) of the Bluetooth service whose index is *n*. (Indexes range from 0 to _btdevicecount.)

Example:

```
echo %@btserviceclassid[0]
```

4.3.4.46 @BTSERVICECOMMENT

@BTSERVICECOMMENT[n] : Returns a comment describing the Bluetooth service whose index is *n*. (Indexes range from 0 to _btdevicecount.)

Example:

```
echo %@btservicecomment[0]
```

4.3.4.47 @BTSERVICENAME

@BTSERVICENAME[n] : Returns the name of the Bluetooth service whose index is *n*. (Indexes range from 0 to _btdevicecount.)

Example:

```
echo %@btservicename[0]
```

4.3.4.48 @BTSERVICEOTHERCLASSID

@BTSERVICEOTHERCLASSID[n] : Returns a list of other class IDs used by the Bluetooth service whose index is *n*. (Indexes range from 0 to _btdevicecount.)

Example:

```
echo %@btserviceotherclassid[0]
```

4.3.4.49 @BTSERVICEPORT

@BTSERVICEPORT[n] : Returns the port used by the Bluetooth service whose index is *n*. (Indexes range from 0 to _btdevicecount.)

Example:

```
echo %@btserviceport[0]
```

4.3.4.50 @BTSERVICEPROTOCOL

@BTSERVICEPROTOCOL[n] : Returns the protocol used by the Bluetooth service whose index is *n*. (Indexes range from 0 to _btdevicecount.)

Example:

```
echo %@btserviceprotocol[0]
```

4.3.4.51 @BWRITE

@BWRITE[*handle,offset,filehandle,fileoffset,length*] : Write from a binary buffer to a file.

handle - a binary handle from @BALLOC

offset - the byte offset in the buffer (decimal or hex)

filehandle - a file handle opened for writing (from @FILEOPEN)

fileoffset - the write offset (from the current file position) (decimal or hex)

length - the number of bytes to write (decimal or hex)

@BWRITE returns the number of bytes written

Example:

```
set fhandle=%@fileopen[filename,w]
set bhandle=%@balloc[128]
set value=%@bwrite[%bhandle,0,%fhandle,0,32]
```

4.3.4.52 @CAPI

@CAPI[module,function[,integer | PINT=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.

4.3.4.53 @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.

Examples:

```
echo %@caps[" ",i love take command]
I Love Take Command

echo %@caps[",",peter,paul,mary]
Peter,Paul,Mary

echo %@caps[" ^"","sacrebleu!", he said]
"Sacrebleu!", He Said
```

4.3.4.54 @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.

If the argument is a partial filename without a drive, @CDROM will expand the filename to get the drive.

Examples:

```
echo %@cdrom[C:]
0

echo %@cdrom[G:]
1
```

4.3.4.55 @CEILING

@CEILING[*n***]** : Returns the value of the smallest integer that is not less than **n**. @CEILING will perform an implicit @EVAL on its argument, so you can enter an arithmetic expression.

Examples:

```
echo %@ceiling[3.14]
4

echo %@ceiling[-3.14]
-3

echo %@ceiling[0]
0

echo %@ceiling[123*37.36]
4596
```

See also: @FLOOR.

4.3.4.56 @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.

To display the non-ASCII Unicode characters (>= 128), you need to be using a Unicode font in *Take Command* and/or *TCC*.

Numeric input may be entered in either decimal format (a sequence of 0-9 digits) 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]
A

echo %@char[65 97 66 98 67 99]
AaBbCc
```

4.3.4.57 @CKSUM

@CKSUM[*filename***] :** Returns a checksum matching the Unix / Linux cksum utility (Posix 100.32 decimal format).

Example:

echo **[%@cksum[tcc.exe]](#)**
2233279932

4.3.4.58 @CLIP

@CLIP[*n[, clipboard]***]** : 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.

The optional second parameter (0-9) specifies the clipboard you want to use (CLIP0: - CLIP9:). The default Windows clipboard is CLIP0:.

Numeric input may be entered in either decimal format (a sequence of 0-9 digits) 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

rem Return the first line in CLIP7:
echo %@clip[0,7]
```

4.3.4.59 @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
Saved to the clipboard
```

4.3.4.60 @CLIPWN

@CLIPWN[*clipboard, string***]** : Writes the ***string*** to the specified clipboard (0 - 9). Returns **0** if the operation was successful.

Examples:

```
if "%@clipwn[2,save this line]" eq "0" echo Saved to CLIP2:
Saved to CLIP2:
```

4.3.4.61 @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 (a sequence of 0-9 digits) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

Examples:

```
echo %@color[]
```
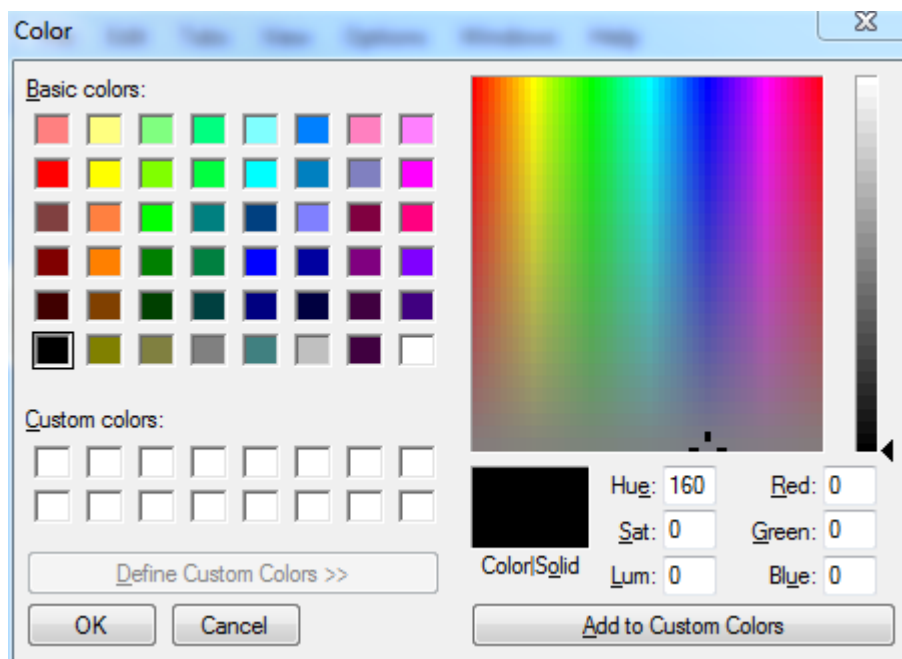


```
echo %@color[155,0,0]
```
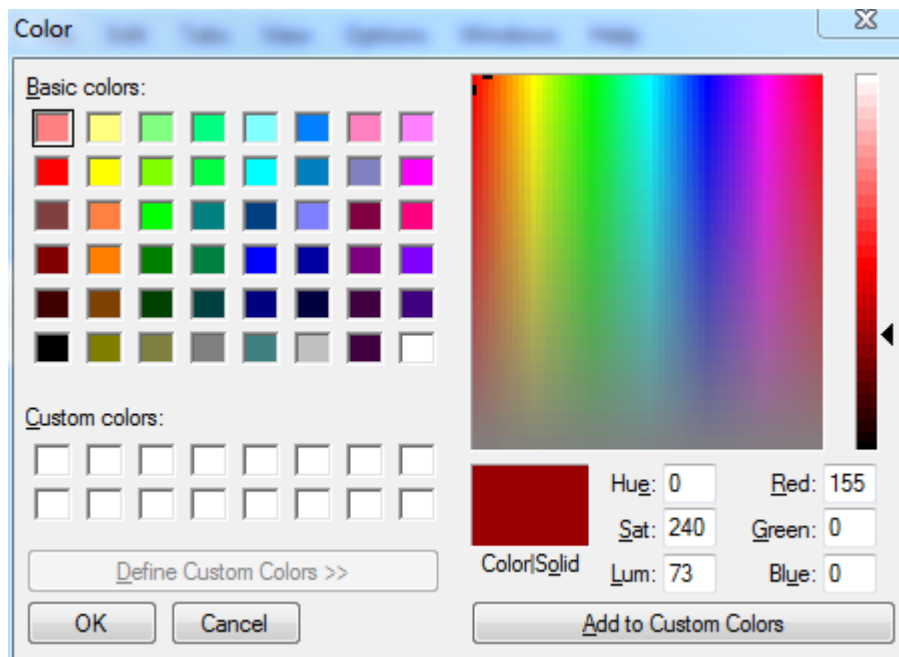
4.3.4.62 @COMMA

@COMMA[*n***]** : Returns the number with the thousands separator inserted where appropriate.

> **Note:** Some variable functions can directly generate a numeric result with appropriate thousand separators if you add a *c* to their scale parameter.

Examples:

```
echo %@comma[12345678]
12,345,678

echo %@comma[0.12345678]
0.12345678

echo %@comma[%_xpixels]
1,920
```

See also: @CONVERT, @FORMAT, @FORMATN.

4.3.4.63 @COMPARE

@COMPARE[*file1,file2***]** : Returns **1** if the two files are identical, or **0** if they differ. @COMPARE supports FTP or HTTP filenames for either *file1* or *file2*, but cannot compare two FTP or HTTP files.

Example:

```
echo %@compare["c:\windows\system32\cmd.exe","c:
\windows\syswow64\cmd.exe"]
0
```

©2024 JP Software, Inc.

4.3.4.64 **@COMPUTERNAME**

@COMPUTERNAME[*n*] - Returns a DNS or NetBIOS name associated with the local computer. The names are created at startup time. The type of name to be retrieved is specified by *n*:

0	The NetBIOS name of the local computer or the cluster associated with the local computer
1	The DNS name of the local computer or the cluster associated with the local computer
2	The name of the DNS domain assigned to the local computer or the cluster associated with the local computer.DNS
3	The fully qualified DNS name that uniquely identifies the local computer or the cluster associated with the local computer
4	The NetBIOS name of the local computer
5	The DNS host name of the local computer
6	The name of the DNS domain assigned to the local computer
7	The fully qualified DNS name that uniquely identifies the computer

Examples:

```
echo %computername[0]
MSI-PC
```

4.3.4.65 **@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.

Example:

```
echo %@console[TCC Prompt]
1
```

4.3.4.66 **@CONSOLEB**

@CONSOLEB[*handle*] - create or restore a console screen buffer. "Handle" is the handle to the desired screen buffer. If "handle is -1, @CONSOLEB just returns the current buffer handle. If "handle" is 0, @CONSOLEB will create and activate a new console screen buffer. If "handle" is non-zero, @CONSOLEB will switch to that screen buffer. @CONSOLEB returns the handle to the active screen buffer. You can close an console handle with the @FILECLOSE function.

@CONSOLEB allows you to preserve the contents of the current screen buffer by switching to a second buffer temporarily and and then back to the original buffer.

Examples:

```
echo %@consoleb[-1]
760

echo %@consoleb[0]
1532

echo %@consoleb[760]
```

4.3.4.67 @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]
binary 1010101 is decimal 85

echo decimal 20 is hex %@convert[10,16,20]
decimal 20 is hex 14

echo hexadecimal FF is octal %@convert[16,8,FF]
hexadecimal FF is octal 377

echo this year is %@convert[10,2,%_year] in binary
this year is 11111011100 in binary
```

See also: @COMMA, @FORMAT, @FORMATN.

4.3.4.68 @COUNT

@COUNT[*c,string***]** : Returns the number of times the character ***c*** appears in ***string***.

Examples:

```
echo %@count[e,Another function example]
3
```

4.3.4.69 @CRC32

String mode: **@CRC32[***s[a|8],string[,start[,length]***]]**
File mode: **@CRC32[***[d,][f,]filename[,start[,length]***]]**
Binary mode: **@CRC32[***[b,]handle[,start[,length]***]]**

Returns the CRC32 value (using the same algorithm as PKZIP or WINZIP) of the characters in ***string***, the contents of the file ***filename***, or the contents of the binary buffer.

If the first parameter is ***s*** for a Unicode UTF16 string, ***sa*** for an ASCII string, or ***s8*** for a UTF8 string, any leading or trailing whitespace characters in ***string*** are included. If the first argument is a ***b***, the ***filename*** argument should be the handle returned by @BALLOC.

If the first argument for file mode is a ***d***, @CRC32 will return the result in decimal (base 10) format. (This is the same format as POSIX 1003.2.) Otherwise, the result is returned in hexadecimal format.

Filename may be specified with or without an optional ***f***. @CRC32 returns **-1** if the file does not exist, or it cannot be read.

Since ***Take Command*** handles all internal strings as Unicode, @CRC32 will return different results for a string and the identical string in an ASCII file.

See also: @SHA256, @SHA384, @SHA512, and @MD5..

Examples:

```
echo %@crc32["C:\windows\explorer.exe"]
3F1E7CFE

echo %@crc32["%comspec"]
F36EB74C

echo %@crc32[d,"%comspec"]
4084119372
```

4.3.4.70 @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.

If the argument is a partial filename without a drive, @CWD will expand the filename to get the drive.

Examples:

```
echo %@cwd[C:]
c:\Windows

echo %@cwd[%_disk:]
D:\release\version14
```

See also: @CWDS.

4.3.4.71 @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.

If the argument is a partial filename without a drive, @CWDS will expand the filename to get the drive.

Examples:

```
echo %@cwds[C:]
c:\Windows\

echo %@cwds[%_disk:]
D:\release\version17\
```

See also: @CWD.

4.3.4.72 @DATE

@DATE[*date[,format]***]** : Returns the number of days since January 1, 1980 for the specified date. See
date formats 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)
5 ISO 8601 yyyy-Www-d
6 ISO 8601 yyyy-ddd

If you don't supply any argument(s), @DATE defaults to using the current date.

Examples:

```
echo %@date[01-01-2012]
11688

echo %@date[2012-01-01,4]
11688

echo %@date[%_date]
11814
```

4.3.4.73 @DATECONV

@DATECONV[*date,format*] - convert a date from the default format to another format. The output formats are:

0 system default
1 USA (mm/dd/yy)
2 European (dd/mm/yy)
3 Japan (yy/mm/dd)
4 ISO 8601 (yyyy-mm-dd)
5 ISO 8601 yyyy-Www-d
6 ISO 8601 yyyy-ddd

Example:

```
echo %@dateconv[5-1-2012,4]
2012-05-01
```

4.3.4.74 @DATEFMT

@DATEFMT[*date,format*] - Formats a date/time in a custom format.

date - The date to format (in yyyy-mm-dd hh:mm:ss format). If *date* is *, @DATEFMT defaults to the current date/time. Valid dates are January 1, 1970 (1970-1-1) to December 31, 3000 (3000-12-31). The time must be in 24-hour format.

format - The custom format to use. (Note that the %'s will normally need to be doubled or escaped to prevent TCC-RT from expanding them before @DATEFMT sees them.) The formatting options are:

Code   Replacement string

%a	Abbreviated weekday name in the locale
%A	Full weekday name in the locale
%b	Abbreviated month name in the locale
%B	Full month name in the locale
%c	Date and time representation in the "C Locale" - equivalent to `"%a %b %e %T %Y"`
%C	The year divided by 100 and truncated to an integer, as a decimal number (00–99)
%d	Day of month as a decimal number (01 - 31)
%D	Equivalent to **%m/%d/%y**
%e	Day of month as a decimal number (1 - 31), where single digits are preceded by a space
%F	Equivalent to **%Y-%m-%d**
%g	The last 2 digits of the ISO 8601 week-based year (00 - 99)
%G	The ISO 8601 week-based year as a decimal number
%h	Abbreviated month name (equivalent to **%b**)
%H	Hour in 24-hour format (00 - 23)
%I	Hour in 12-hour format (01 - 12)
%j	Day of the year as a decimal number (001 - 366)
%m	Month as a decimal number (01 - 12)
%M	Minute as a decimal number (00 - 59)
%n	A newline character (**\n**)
%p	The locale's A.M./P.M. indicator for 12-hour clock
%r	The locale's 12-hour clock time
%R	Equivalent to **%H:%M**
%S	Second as a decimal number (00 - 59)
%t	A horizontal tab character (**\t**)
%T	Equivalent to **%H:%M:%S**, the ISO 8601 time format
%u	ISO 8601 weekday as a decimal number (1 - 7; Monday is 1)
%U	Week number of the year as a decimal number (00 - 53), where the first Sunday is the first day of week 1
%V	ISO 8601 week number as a decimal number (00 - 53)
%w	Weekday as a decimal number (0 - 6; Sunday is 0)
%W	Week number of the year as a decimal number (00 - 53), where the first Monday is the first day of week 1
%x	Date representation for the locale
%X	Time representation for the locale
%y	Year without century, as decimal number (00 - 99)
%Y	Year with century, as decimal number
%z	The offset from UTC in ISO 8601 format; no characters if time zone is unknown
%Z	Either the locale's time-zone name or time zone abbreviation, depending on registry settings; no characters if time zone is unknown
%%	Percent sign

Characters that do not begin with a % are displayed unchanged.

The **#** flag may prefix any formatting code. In that case, the meaning of the format code is changed as follows:

Format code	Meaning
%#a, **%#A**, **%#b**, **%#B**, **%#g**, **%#G**, **%#h**, **%#n**, **%#p**, **%#t**, **%#u**, **%#w**, **%#X**, **%#z**, **%#Z**, **%#%**	**#** flag is ignored.
%#c	Long date and time representation, appropriate for the locale. For example: "Tuesday, February 25, 2020, 12:41:29".
%#x	Long date representation, appropriate to the locale. For example: "Tuesday, February 25, 2020".
%#d, **%#D**, **%#e**, **%#F**, **%#H**, **%#I**, **%#j**, **%#m**, **%#M**, **%#r**, **%#R**, **%#S**, **%#T**, **%#U**, **%#V**, **%#W**, **%#y**, **%#Y**	Remove leading zeros or spaces (if any).

The ISO 8601 week and week-based year produced by **%V**, **%g**, and **%G**, uses a week that begins on Monday, where week 1 is the week that contains January 4th, which is the first week that includes at least four days of the year. If the first Monday of the year is the 2nd, 3rd, or 4th, the preceding days are part of the last week of the preceding year. For those days, **%V** is replaced by 53, and both **%g** and **%G** are replaced by the digits of the preceding year.

4.3.4.75 @DAY

@DAY[_date[,format]_**]** : Returns the numeric day of the month for the specified date. See date formats 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)
- **5** ISO 8601 yyyy-Www-d
- **6** ISO 8601 yyyy-ddd

Examples:

```
echo %@day[2012-01-01]
1

echo %@day[%_date]
6
```

4.3.4.76 @DEBUG

@DEBUG[*string***]** : Write a string to the system debugger.

4.3.4.77 @DEC

@DEC[*string***]** : Returns :

- **-1** if ***string*** is empty
- otherwise the same value as @EVAL[*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]
```

Example:

```
set start=5
set result=%@dec[start]
echo %result
4
```

4.3.4.78 @DECIMAL

@DECIMAL[*number***]**: Returns the portion of ***number*** to the right of the decimal character as an integer numeric string. Trailing zeros are used to pad to the minimum precision specified for @EVAL. 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

4.3.4.79 @DESCRIPT

@DESCRIPT[*filename***]**: Returns the file description for the specified filename (see DESCRIBE). If there is no description for the file, @DESCRIPT returns an empty string.

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"]
```

4.3.4.80 @DEVICE

@DEVICE[name**]** : Returns **1** if the specified name is a character device (such as a serial port), or **0** if not. A trailing **:** is optional except for the pseudo-device CLIP: (to differentiate it from a possible filename named "clip").

Examples:

```
echo %@device[%comspec]
0

echo %@device[lpt1]
1

echo %@device[com1]
1

echo %@device[com5]
0

echo %@device[clip]
0

echo %@device[clip:]
1
```

4.3.4.81 @DIGITS

@DIGITS[n**]**: Returns **1** if the string is composed of decimal digits only, otherwise it returns **0.** The decimal character, the thousands character, and the sign characters (+ or -) are not digits, and if they are present in the string @DIGITS will return **0**.

Examples:

```
echo %@digits[12345]
1

echo %@digits[-12345]
0

echo %@digits[1.2345]
0
```

4.3.4.82 @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 / POPD.

Numeric input may be entered in either decimal format (a sequence of 0-9 digits) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

See also: DIRS, POPD, and PUSHD

Examples:

```
pushd c:\windows
pushd c:\windows\system32
echo %@dirstack[0]
C:\

echo %@dirstack[1]
C:\Windows

echo %@dirstack[]
2
```

4.3.4.83 @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). If the scale specification is suffixed with ***c*** the result will be formatted using the thousands separator.

@DISKFREE supports OpenAFS names.

See also: @DISKTOTAL and @DISKUSED.

Examples:

```
echo %@diskfree[c:]
19941240832

echo %@diskfree[%_disk:,Kc]
503,709,632
```

4.3.4.84 @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). If the scale specification is suffixed with *c* the result will be formatted using the thousands separator.

@DISKTOTAL supports OpenAFS names.

See also: @DISKFREE and @DISKUSED.

Examples:

```
echo %@disktotal[c:]
120031539200

echo %@disktotal[%_disk:,Kc]
976,657,404
```

4.3.4.85 @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). If the scale specification is suffixed with *c* the result will be formatted using the thousands separator.

@DISKUSED supports OpenAFS names.

See also: @DISKFREE and @DISKTOTAL.

Examples:

```
echo %@diskused[c:]
100090298368

echo %@diskused[%_disk:,Kc]
472,947,772
```

4.3.4.86 @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.

4.3.4.87 @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 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)
5   ISO 8601 yyyy-Www-d
6   ISO 8601 yyyy-ddd

Examples:

```
echo %@dow[01-01-1980]
Tue


echo %@dow[%_date]
Sun
```

See also: @IDOW.

4.3.4.88 @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 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)
5   ISO 8601 yyyy-Www-d
6   ISO 8601 yyyy-ddd

Examples:

```
echo %@dowf[01-01-1980]
Tuesday

echo %@dowf[%_date]
Sunday
```

See also: @IDOWF.

4.3.4.89 @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 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)
5　ISO 8601 yyyy-Www-d
6　ISO 8601 yyyy-ddd

Examples:

```
echo %@dowi[01-01-1980]
3

echo %@dowi[%_date]
1
```

4.3.4.90 @DOY

@DOY[_date[,format]_**]** : Returns the day of year (1 - 366) for the specified date. See date formats 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)
5　ISO 8601 yyyy-Www-d
6　ISO 8601 yyyy-ddd

Examples:

```
echo %@doy[02-02-2010]
33

echo %@doy[%_date]
127
```

4.3.4.91 @DRIVE

@DRIVE[_filename_**]**: Returns the drive of **_filename_**. If the **_filename_** parameter doesn't contain a drive specification, @DRIVE will expand **_filename_** before the drive is extracted.

filename must be in quotes if it contains white space or special characters.

If the path is a UNC, @DRIVE will return the computer name + sharename.

Examples:

```
echo @drive["c:\program files\xyz.abc"]
c:

echo "%@drive[\\system1\d_drive\myfile]"
\\system1\d_drive
```

4.3.4.92 @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

Examples:

```
echo %@drivetype[c:]
3

echo %@drivetype[z:]
4

echo %@drivetype[e:]
5
```

4.3.4.93 @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

Examples:

```
echo %@drivetypeex[c:]
3
```

```
echo %@drivetypeex[z:]
4

echo %@drivetypeex[e:]
7
```

4.3.4.94 @EMAIL

@EMAIL[*address*] : Validate an email address.

@EMAIL uses the regular expression "^[\w-]+(\.[\w-]+)*@([a-z0-9-]+(\.[a-z0-9-]+)*?\.[a-z]{2,6}|(\d{1,3}\.){3}\d{1,3})(:\d{4})?$" to validate the address. This matches 99.99% of valid email address including ip's (which are rarely used). Allows for a-z0-9_.- in the username, but not ending in a full stop (i.e user.@domain.com is invalid) and a-z0-9- as the optional sub domain(s) with domain name and a 2-7 char (a-z) tld.

```
echo %@email[bob@jpsoft.com]
1

echo %@email[bob!@jpsoft.com]
0
```

4.3.4.95 @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](). Returns a null string if there are fewer than ***n-1*** 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,L*]
\\LINKSTATION

for %i in (0 1 2) echo %@enumservers[%i,*]
\\LINKSTATION
\\MUSIC
\\WEBHOST
```

4.3.4.96 @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](). Returns a null string If there are fewer than ***n-1*** 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,\\LINKSTATION\*]
\\LINKSTATION\info

for %i in (0 1 2) echo %@enumshares[%i,\\LINKSTATION\*]
\\LINKSTATION\info
\\LINKSTATION\share
\\LINKSTATION\archive
```

4.3.4.97 @ERRTEXT

@ERRTEXT[*n***]** : Returns the operating system error text for the specified code. The text will be in the default language.

Examples:

```
echo %@errtext[2]
The system cannot find the file specified.

echo %@errtext[255]
The extended attributes are inconsistent.

echo %@errtext[%_syserr]
Incorrect function.
```

4.3.4.98 @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 below describes what ***expression*** may contain. Display precision and output format 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 2,147,483,647 digits in Windows x64. (Windows x86 will be limited by memory to much less). If you want to use more than the default decimal values you'll need to change your @Eval Precision configuration options or use the "=x.y" format in @EVAL. The integer-only operators (AND, OR, and XOR) are limited to 64-bit integers.

- ▸ Parameter Interpretation
- ▸ Arithmetic operators
- ▸ Trigonometric and transcendental functions
- ▸ Other functions
- ▸ Order of precedence
- ▸ Precision of internal calculations
- ▸ Display precision and output format
- ▸ Examples

Parameter Interpretation

Expression may contain environment and internal variables, array 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 or Other functions, connected by Arithmetic operators and optionally grouped with parentheses.

@EVAL permits you to simplify ***expression*** by dropping the `%` percent mark in front of the names of environment variables. This also prevents the TCC-RT parser from expanding (possibly erroneously) variables before passing them to @EVAL. You must include `%` for internal variables and variable functions. @EVAL also permits you to use characters which normally have special meaning for TCC-RT 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 accepts numbers in the scientific notation exponent syntax; i.e. 1575e-2 = 15.75. You can specify scientific notation output with the syntax **@eval[...=E]**. For example:

```
echo %@eval[1.4567e+4*7.6541e+2=E]
```

You can combine =E with a display precision (see below):

```
echo %@eval[1.4567e+4*7.6541e+2=E1.20]
```

Number base

If a string starts with the characters `0x` it is interpreted as an integer in hexadecimal notation. If a string starts with the characters **0b** it is interpreted as an integer in binary notation. 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 function.

You can specify binary output with the special syntax `@eval[...=B]`. For example:

```
echo %@eval[3*6=B]
```

Hex and binary output is limited to 64-bit (signed) integers.

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
!	boolean not

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**
>	greater than
<	less than

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
log2(x)	binary 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*.

Other functions

abs(x)	absolute value
ceil(x)	ceiling
fact(x)	factorial
floor(x)	floor
gcd(x y)	greatest common divisor (max 64-bit integer)
lcm(x y)	least common multiple (max 64-bit integer)
ror(x y z)	rotate **x** right **y** bits with a variable size of **z** (in bits) (max 64-bit integer)
rol(x y z)	rotate **x** left **y** bits with a variable size of **z** (in bits) (max 64-bit integer)

Order of precedence

1. variables
2. expressions in matching parentheses
3. functions listed in Trigonometric and transcendental functions
4. exponentiation
5. multiplication, division, and MOD
6. addition and subtraction
7. >, <, AND, OR, XOR, NOT, 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 30,000 digits; it is highly unlikely you'll need greater precision than this! A few functions (gcd, lcm, ror, rol) use 64-bit integers.

Display precision and output format

The maximum display precision is 15,000 digits to the left of the decimal point and 15,000 digits to the right. You can alter the default decimal precision with the OPTION command, the @EVAL Precision configuration options, and with the SETDOS /F command. You can change the decimal separator with the decimal character configuration option or the SETDOS /G command.

You can alter the display format for the current instance of @EVAL by specifying *displayformat*.

Scientific notation display format

If *displayformat* is **E**, output will be in scientific notation. For example:

```
echo %@eval[1.4567e+4*7.6541e+2=E]
```

You can combine =E with a display precision (see below):

```
echo %@eval[1.4567e+4*7.6541e+2=E1.15]
```

Hexadecimal display format

If *displayformat* is the letter **H**, output will be hexadecimal. If *displayformat* is **X,** the output will be hexadecimal with a leading **0x**.

Binary display format

If *displayformat* is the letter **B**, output will be binary.

Explicit precision

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.**

If the last character of the *displayformat* is **+**, @EVAL will prefix positive numbers with a **+**.

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`
`@eval[3 / 7=2+]`	`+0.42`

See also: @DEC and @INC.

4.3.4.99 @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 ? and _? 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.

4.3.4.100 @EXECARRAY

@EXECARRAY[*array,command***]** : Execute the specified command and store the resulting lines in the specified array variable. The array must be one-dimensional.

You must define the array before running @EXECARRAY. For example:

```
PROMPT=%@exec[@color 15 on %@if[%@removable[%_disk] eq 0,2,4] & echos [%
_cwd%] & color 11 on 0]$s
```

@EXECARRAY will read the number of lines specified in the array size definition, or the number of lines in the command output (whichever is less). @EXECARRAY returns the return value of the command.

The number of lines stored in the array is saved in the _EXECARRAY internal variable.

4.3.4.101 @EXECSTR

@EXECSTR[*[n,]command***]** : Runs the specified ***command*** and returns line *n* (or the first line if *n* is not specified) written to stdout by ***command***. For example, to return the third line returned by VER /R:

```
echo %@execstr[2,ver /r]
```

If *n* is negative, @EXECSTR starts at the last line and counts backwards.

@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:-d /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.

See also: @EXEC and @EXECARRAY.

4.3.4.102 @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
9	Windows x64 GUI
10	Windows x64 console
11	EFI
12	EFI boot driver
13	EFI runtime driver
14	EFI ROM
15	XBox
16	Windows boot application

Examples:

```
echo %@exetype["dc:\windows\explorer.exe"]
6

echo %@exetype["%comspec"]
7
```

4.3.4.103 @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 <u>wildcards</u> and <u>include lists</u>. Returns an empty string if no files match. 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 <u>File Selection</u>. Multiple ***range*** parameters may be included, but not more than one each of <u>description range</u>, <u>size range</u>, <u>date range</u>, and <u>time range</u>. ***Range*** parameters must precede ***filename***.

Examples:

```
echo %@expand[/[s2k,3k] *.txt]
```

> displays all files with extension **txt** in the current directory with size at least 2000 bytes and at most 3000 bytes

```
echo %@expand[*,d]
```

> displays all subdirectories

```
echo %@expand[/[d-365] %windir\w*.exe;w*.dll]
```

> displays all files at most 365 days old in the Windows directory, with extension *EXE* or *DLL*, and name beginning with **W.**

4.3.4.104 @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:

```
set COMSPEC="c:\program files\jpsoft\tcmd28\tcc.exe"
echo %@ext[%comspec]
exe

echo %@ext["LFN Names may have.very long extensions"]
very long extensions
```

4.3.4.105 @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, @FIELDS, @WORD and @WORDS 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, 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 and @FIELDS differ from @WORD and @WORDS in how multiple consecutive separators are counted. @WORD and @WORDS consider a sequence as a single separator, and ignore separators at either end of string. In contrast, @FIELD and @FIELDS count each occurrence of a separator individually, including those at either end of string.

Numeric input may be entered in either decimal format (a sequence of 0-9 digits) 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, @WORDS, @FIELDS.

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

4.3.4.106 @FIELDS

@FIELDS[["sep_list",]string**]** : Returns the number of fields in **string**.

The default list of separators for @FIELD, @FIELDS, @WORD and @WORDS 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, 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 and @FIELDS differ from @WORD and @WORDS in how multiple consecutive separators are counted. @WORD and @WORDS consider a sequence as a single separator, and ignore separators at either end of string. In contrast, @FIELD and @FIELDS count each occurrence of a separator individually, including those at either end of string.

If **string** is double quoted, you must specify **sep_list**.

Example:

```
echo %@fields[" ,",Now is the time]
4
```

See also: @WORD, @WORDS, @FIELD.

4.3.4.107 @FILEAGE

@FILEARRAY[*array,filename*] : Read the specified file and store the resulting lines in the specified array variable (one line per element). The array must be one-dimensional.

You must define the array before running @FILEARRAY. For example:

```
setarray aresult[10]
echo %@filearray[aresult,test.dat]
```

@FILEARRAY will read the number of lines specified in the array size definition, or the number of lines in the files (whichever is less).

@FILEARRAY will return the number of lines read.

4.3.4.108 @FILEARRAY

@FILEARRAY[*array,filename*] : Read the specified file and store the resulting lines in the specified array variable (one line per element). The array must be one-dimensional.

You must define the array before running @FILEARRAY. For example:

```
setarray aresult[10]
echo %@filearray[aresult,test.dat]
```

@FILEARRAY will read the number of lines specified in the array size definition, or the number of lines in the files (whichever is less).

@FILEARRAY will return the number of lines read.

@FILEARRAY supports the TCC-RT clipboards (CLIP0: - CLIP9:).

4.3.4.109 @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! If you use it with any other number you may damage other files opened by TCC-RT (or by the program which started TCC-RT).

See also the related handle-based functions:

@FILEOPEN	Open a file handle
@FILEREAD	Read next line from a file handle
@FILESEEK	Move a file handle pointer
@FILESEEKL	Move a file handle pointer to a specified line
@FILEWRITE	Write next line to a file handle
@FILEWRITEB	Write data to a file handle
@TRUNCATE	Truncate the file at the current position of the file handle pointer.

Examples:

```
set h=%@fileopen["d:\path\myfile.txt",write]
echo writing %@filewrite[%h,this is a test] bytes
echo closing handle #%h: %@fileclose[%h]
```

4.3.4.110 @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). **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.

@FILETIME supports HTTP & HTTPS filenames, for last write only. Wildcards are not supported (HTTP limitation).

Example:

```
echo %@filedate["%comspec",c,4]
2012-04-29
```

See Time Stamps, @FILETIME, @FILEAGE.

4.3.4.111 @FILEHANDLE

@FILEHANDLE[*handle*] : Returns the filename for the specified file handle (opened with @FILEOPEN).

Example:

```
set h=%@fileopen["d:\path\myfile.txt",r]
echo handle %h is : %@filehandle[%h]
handle 756 is : d:\path\myfile.txt
```

4.3.4.112 @FILELOCK

@FILELOCK[*filename*] : Returns the PIDs of the processes with a lock on the specified file.

Example:

```
echo %@filelock[d:\path\myfile.txt]
```

4.3.4.113 @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"]
myfile.exe

echo %@filename["%comspec"]
tcc.exe
```

4.3.4.114 @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 without a ***length***, and to write text using @FILEWRITE. Binary mode should be used to read binary data with @FILEREAD with a ***length***, and to write binary data with @FILEWRITEB. Returns -1 if the file cannot be opened.

Filename must be in quotes if it contains white space or special characters. To read from standard input, use **CON:** for the filename.

To open a file for both reading and writing, open it in append mode, then use @FILESEEK 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. @FILEOPEN will not return until another process connects to the pipe. 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 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-RT.

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	Close a file handle
@FILEREAD	Read next line from a file handle
@FILESEEK	Move a file handle pointer
@FILESEEKL	Move a file handle pointer to a specified line
@FILEWRITE	Write next line to a file handle
@FILEWRITEB	Write data to a file handle
@TRUNCATE	Truncate the file at the current position of the file handle pointer.

4.3.4.115 @FILEREAD

@FILEREAD[_n[,length]_**]** : Reads data from the file whose handle is _**n**_. Returns the string ****EOF**** if you attempt to read past the end of the file. If _**length**_ is not specified, **@FILEREAD** will read until the next CR or LF (end of line) character. If _**length**_ is specified, @FILEREAD will read _**length**_ bytes regardless of any end of line characters.

Numeric input may be entered in either decimal format (a sequence of 0-9 digits) 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. If you use it with any other number you may damage other files opened by TCC-RT (or by the program which started TCC-RT).

Beware of characters with special meaning to _**Take Command**_, such as redirection and piping symbols, within the file. Use SETDOS /X with appropriate codes as needed.

Example:

```
set h=%@fileopen["d:\path\myfile.txt",r]
echo reading %@fileread[%h,32]
echo closing handle #%h: %@fileclose[%h]
```

See also the related handle-based functions:

@FILECLOSE	Close a file handle
@FILEOPEN	Open a file handle
@FILESEEK	Move a file handle pointer
@FILESEEKL	Move a file handle pointer to a specified line
@FILEWRITE	Write next line to a file handle
@FILEWRITEB	Write data to a file handle

@TRUNCATE   Truncate the file at the current position of the file handle pointer.

4.3.4.116 @FILEREADB

@FILEREADB[*n,length[,h x]***]** : Reads *length* 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.

The optional third parameter (**h** or **x**) species the output format:

    h   Output is 2-digit hex (00 - FF)
    x   Output is 0x00 - 0xFF)

Numeric input may be entered in either decimal format (a sequence of 0-9 digits) 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. If you use it with any other number you may damage other files opened by TCC-RT (or by the program which started TCC-RT).

Beware of characters with special meaning to TCC-RT, such as redirection and piping symbols, within the file. Use SETDOS /X with appropriate codes as needed.

Example:

```
set h=%@fileopen["d:\path\myfile.txt",r]
echo reading %@filereadb[%h,32]
echo closing handle #%h: %@fileclose[%h]
```

See also the related handle-based functions:

    @FILECLOSE   Close a file handle
    @FILEOPEN   Open a file handle
    @FILESEEK   Move a file handle pointer
    @FILESEEKL   Move a file handle pointer to a specified line
    @FILEWRITE   Write next line to a file handle
    @FILEWRITEB   Write data to a file handle
    @TRUNCATE   Truncate the file at the current position of the file handle pointer.

4.3.4.117 @FILES

@FILES[*/S[[+]n]] [range...][/H] filename[,[{+|-}]rhsadecijopt]***]** : Returns the number of files within *range* that match *filename* and have the specified attributes. *Filename* may contain wildcards and include lists. Returns 0 if no files match. To check files in multiple directories use @FILES once for each, and add the results with @EVAL.

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. Multiple *range* parameters may be included, but not more than one each of description range, size range, date range, and time range. *Range* parameters must precede *filename*. Exclusion ranges 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 specify a **+** followed by a number after the /S, @FILES will not count any files until it gets to that depth in the subdirectory tree. For example, if you have a directory tree \a\b\c\d\e, /S+2 will not count anything in \a or \a\b.

If you include the optional **/H** argument, @FILES will not include the "." and ".." directory entries in the count.

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**

4.3.4.118 @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**.

start	reference location
0	beginning of file
1	current file pointer
2	end of file

Numeric input may be entered in either decimal format (a sequence of 0-9 digits) 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. If you use it with any other number you may damage other files opened by TCC-RT (or by the program which started TCC-RT).

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.

Example:

```
set h=%@fileopen["d:\path\myfile.txt",rw]
echo file size = %@fileseek[%h,0,2]
echo closing handle #%h: %@fileclose[%h]
```

See also the related handle-based functions:

@FILECLOSE	Close a file handle
@FILEOPEN	Open a file handle
@FILEREAD	Read next line from a file handle
@FILESEEKL	Move a file handle pointer to a specified line
@FILEWRITE	Write next line to a file handle
@FILEWRITEB	Write data to a file handle
@TRUNCATE	Truncate the file at the current position of the file handle pointer.

4.3.4.119 @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 (a sequence of 0-9 digits) 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. If you use it with any other number you may damage other files opened by TCC-RT (or by the program which started TCC-RT).

Example:

```
set h=%@fileopen["d:\path\myfile.txt",rw]
echo file line 10 = %@fileseekl[%h,10,2]
echo closing handle #%h: %@fileclose[%h]
```

See also the related handle-based functions:

@FILECLOSE	Close a file handle
@FILEOPEN	Open a file handle
@FILEREAD	Read next line from a file handle
@FILESEEK	Move a file handle pointer
@FILEWRITE	Write next line to a file handle
@FILEWRITEB	Write data to a file handle

@TRUNCATE   Truncate the file at the current position of the file handle pointer.

4.3.4.120 @FILESIZE

@FILESIZE[*[/S[[+]n]] [range...] filename[,[scale[c][,a]]***]** : Returns the size of a file, or **-1** if the file does not exist. If **filename** includes wildcards or an include list, 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). Adding the letter **c** requests the result be formatted using the thousands separator.

The optional **range** parameter defines properties of the files that will be included in the result as specified in File Selection. Multiple **range** parameters may be included, but not more than one each of description range, size range, date range, and time range. **Range** parameters must precede **filename**. Exclusion ranges 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.

If you specify a **+** followed by a number after the /S, @FILESIZE will not count any file sizes until it gets to that depth in the subdirectory tree. For example, if you have a directory tree \a\b\c\d\e, /S+2 will not count anything in \a or \a\b.

@FILESIZE supports returning the size of file streams. @FILESIZE also supports retrieving sizes for HTTP and HTTPS files. (Note that due to HTTP protocol limitations, you cannot use wildcards or scan subdirectories.)

Examples:

```
echo %@filesize[d:\path\myfile.ext]
417

echo %@filesize["%comspec",bc]
359,400

echo %@filesize["%comspec",bc,a]
360,448
```

4.3.4.121 @FILETIME

@FILETIME[*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 for additional details).

If you append a *u* to the second argument, @FILETIME will display the time in UTC.

@FILETIME supports HTTP & HTTPS filenames, for last write only. Wildcards are not supported (HTTP limitation).

Examples:

```
echo %@filetime["D:\my path\myfile.exe"]
16:40

echo %@filetime["%comspec",c,s]
11:01:40
```

See also: @FILEDATE, @FILEAGE.

4.3.4.122 @FILETYPE

@FILETYPE[filename] - returns the encoding type of the file.

You must enable UTF8 input for TCC-RT to recognize UTF8 files; see OPTION / Setup.

The possible return values are:

> ASCII
> UTF8
> UTF16

4.3.4.123 @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 (a sequence of 0-9 digits) 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 If you use it with any other number you may damage other files opened by TCC-RT (or by the program which started *TCC-RT*).

Beware of characters with special meaning to *TCC-RT*, such as redirection and piping symbols, within the file. Use SETDOS /X with appropriate codes as needed.

Example:

```
set h=%@fileopen["d:\path\myfile.txt",w]
echo writing %@filewrite[%h,32]
echo closing handle #%h: %@fileclose[%h]
```

See also the related handle-based functions:

@FILECLOSE	Close a file handle
@FILEOPEN	Open a file handle

[@FILEREAD](#)	Read next line from a file handle
[@FILESEEK](#)	Move a file handle pointer
[@FILESEEKL](#)	Move a file handle pointer to a specified line
[@FILEWRITEB](#)	Write data to a file handle
[@TRUNCATE](#)	Truncate the file at the current position of the file handle pointer.

4.3.4.124 @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 (a sequence of 0-9 digits) 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!](#) If you use it with any other number you may damage other files opened by TCC-RT (or by the program which started **TCC-RT**).

Beware of characters with special meaning to **TCC-RT**, such as redirection and piping symbols, within the file. Use [SETDOS](#) /X with appropriate codes as needed.

Example:

```
set h=%@fileopen["d:\path\myfile.txt",r]
echo writing %@filewriteb[%h,10,Write some characters from this string]
echo closing handle #%h: %@fileclose[%h]
```

See also the related handle-based functions:

[@FILECLOSE](#)	Close a file handle
[@FILEOPEN](#)	Open a file handle
[@FILEREAD](#)	Read next line from a file handle
[@FILESEEK](#)	Move a file handle pointer
[@FILESEEKL](#)	Move a file handle pointer to a specified line
[@FILEWRITE](#)	Write next line to a file handle
[@TRUNCATE](#)	Truncate the file at the current position of the file handle pointer

4.3.4.125 @FILTER

@FILTER[_chars,string_**]** : Removes any characters in "string" that aren't in "chars".

Example:

To remove all non-numeric characters from a variable:

```
set var=abc1234
echo %@filter[0123456789,%var]
1234
```

4.3.4.126 @FINDCLOSE

@FINDCLOSE[_filename_**]**: Signals the end of a @FINDFIRST ... @FINDNEXT 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 ...@FINDNEXT sequence is in effect, a non-zero value otherwise.

Examples:

```
echo %@findfirst[*.exe]
echo %@findclose[]
```

4.3.4.127 @FINDFIRST

@FINDFIRST[_[range...] filename[,[+|-]rhsadecijopt]_**]** : Returns the name of the first file that matches **_filename_**, which may include wildcards and/or an include list, and which file has the properties specified in the optional range and attribute parameters.

Filename must be in quotes if it contains white space or special characters.

If **_filename_** is quoted, the returned filename will also be quoted (if necessary).

The **_range_** and attribute parameters, if included, define properties of the files that will be included in the search as specified in File Selection Multiple **_range_** parameters may be included, but not more than one each of size, date, time, and file exclusion. **_Range_** parameters must precede **_filename_**. Each **_range_** parameter is of the form

```
    /[a...]
```

where **_a_** is one of **_d_**, **_s_**, **_t_**, and/or **_!_**, followed by the range parameters.

On an LFN drive, the returned filename may contain white space or other special characters. Unlike @EXPAND[], 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 for additional details.

@FINDFIRST[] locates the _first_ file matching the requirements. To find more matching files, you must use @FINDNEXT[], and terminate the search with @FINDCLOSE[].

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.

4.3.4.128 @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.

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.

If **filename** is quoted, the returned filename will also be quoted (if necessary).

The attribute parameter, if included, defines the attributes of the files that will be included in the search as specified in Attribute Switches.

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[], 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 for additional details.

@FINDFIRST[] locates the first file matching the requirements. To find more matching files, you must use @FINDNEXT[], and terminate the search with @FINDCLOSE[].

Examples:

```
echo %@findfirst[*]
echo %@findnext[]
echo %@findnext[*,d]
echo %@findclose[]
```

4.3.4.129 @FLOOR

@FLOOR[*n***]**: Returns the largest integer that is not greater than **n**. @FLOOR will perform an implicit @EVAL on its argument, so you can enter an arithmetic expression.

Examples:

```
echo %@floor[3.14]
3

echo %@floor[-3.14]
-4

echo %@floor[0]
0

echo %@floor[123]
```

```
123
```

See also: [@CEILING](#).

4.3.4.130 @FOLDERS

@FOLDERS[[*/S[[+]n]] [range...] dirname[,[{+|-}]rhsadecijopt]***]** : Returns the number of folders (subdirectories) within **range** that match **dirname** and have the specified attributes. Returns 0 if no folders match. To check folders in multiple source directories use @FOLDERS once for each, and add the results with [@EVAL](#).

Dirname must be in double quotes if it contains white space or special characters.

The **range** and [attribute](#) parameters, if included, define properties of the folders that will be included in the result as specified in [File Selection](#). Multiple **range** parameters may be included, but not more than one each of [description range](#), [date range](#), and [time range](#). **Range** parameters must precede **dirname**. [Exclusion ranges](#) are not supported.

If you include the optional */S* argument, @FOLDERS will search the current directory and all of its subdirectories for matching folders. If you specify a number after the /S, @FOLDERS 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 specify a **+** followed by a number after the /S, @FOLDERS will not count any folders until it gets to that depth in the subdirectory tree. For example, if you have a directory tree \a\b\c\d\e, /S+2 will not count anything in \a or \a\b.

If you are searching for subdirectories (i.e., by specifying "d" in the attribute argument), @FOLDERS will not count the "." and ".." directory entries.

Example:

```
echo %@folders[c:\windows]
58

echo %@folders[/s,c:\windows]
17728
```

4.3.4.131 @FONT

@FONT[*n***]** : Returns console font information. *n* is the type of information requested*:*

0 - font name (Windows usually returns an empty string unless you've previously set the font)
1 - font width
2 - font height
3 - font weight
4 - font family
5 - font index in console font table

Examples:

```
echo %@font[0]
Consolas
```

```
echo %@font[1]
6

echo %@font[2]
12

echo %@font[3]
700

echo %@font[4]
54

echo %@font[5]
6
```

4.3.4.132 @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 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.

If **string** doesn't exist, @FORMAT treats it as an empty string and pads the output accordingly.

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, @CONVERT, @FORMATN.

4.3.4.133 @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".)

@FORMATN will use the decimal character for the default locale.

Examples:

```
echo %@formatn[5.10,%@eval[2300*4.7]]
10810.0000000000

echo %@formatn[010.3,5]
000005.000
```

See also: @COMMA, @CONVERT, @FORMAT, @FORMATNC.

4.3.4.134 @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.

Examples:

```
echo %@formatnc[5.10,%@eval[2300*4.7]]
10,810.0000000000

echo %@formatnc[010.3,5]
000005.000
```

See also: @COMMA, @CONVERT, @FORMAT, @FORMATN.

4.3.4.135 @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.

If the argument is a partial filename without a drive, @FSTYPE will expand the filename to get the drive.

Examples:

```
echo %@fstype[c:]
NTFS

echo %@fstype[e:]
FAT32

echo %@fstype[\\Music\iTunes]
NTFS
```

4.3.4.136 @FTYPE

@FTYPE[*xxx[,u]***]** : Returns the open command string for the specified file type. @FTYPE looks first in ...\SHELL\OPEN2\COMMAND, then (if no match was found) in ...\SHELL\OPEN\COMMAND. If the optional second argument **u** is specified, @FTYPE will look in HKCU\SOFTWARE\CLASSES.

Example:

```
echo %@ftype[Word.Document.8]
"C:\Program Files\Microsoft Office\Office14\WINWORD.EXE" /n ""
```

See also @ASSOC and FTYPE.

4.3.4.137 @FULL

@FULL[*filename[,path]***]** : 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 for additional details.

If **filename** is quoted, the returned filename will also be quoted (if necessary).

The optional *path* argument specifies the path name to use to create the name. The path can include relative path operators like "...\".

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:

```
cdd c:\windows
echo %@full[explorer.exe]
C:\Windows\explorer.exe

echo "%@full[.]"
"C:\Windows"

echo "%@full["\Program Files"]"
"C:\Program Files"
```

4.3.4.138 @FUNCTION

@FUNCTION[*name***]** : Returns the definition of the specified user-defined function **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. 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](#) command.

4.3.4.139 @GETDATE

@GETDATE[[*date*]**]** : Display a calendar dialog and returns the selected date in *yyyy-mm-dd* format.

You can optionally pass a default date (also in *yyyy-mm-dd* format). If you do not specify a default date, **@GETDATE** will use the current date.

Example:

```
echo %@getdate[]
```
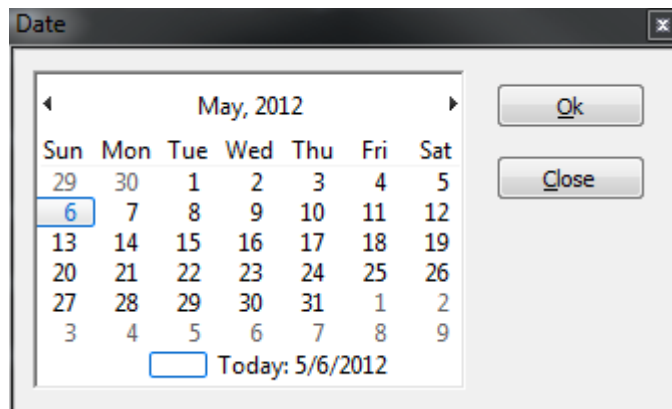

4.3.4.140 @GETDATETIME

@GETDATETIME[[*date time*]**]** :Display a date/time picker dialog and returns the selected date in *yyyy-mm-dd hh:mm:ss* format.

You can optionally pass a default date and time (also in *yyyy-mm-dd hh:mm:ss* format). If you do not specify a default date, **@GETDATETIME** will use the current date and time.

Example:

```
echo %@getdatetime[]
```
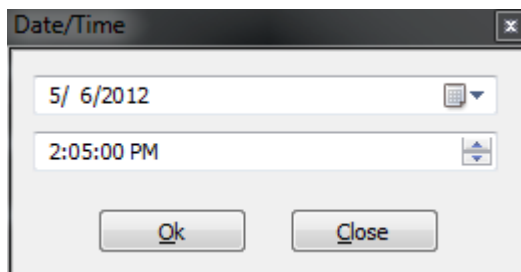

4.3.4.141 @GETDIR

@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 for additional details.

If *path* is quoted, the returned filename will also be quoted (if necessary).

@GETDIR accepts an optional second parameter to set the title of the dialog box.

Example:

```
cdd %@getdir["C:\windows"]
```
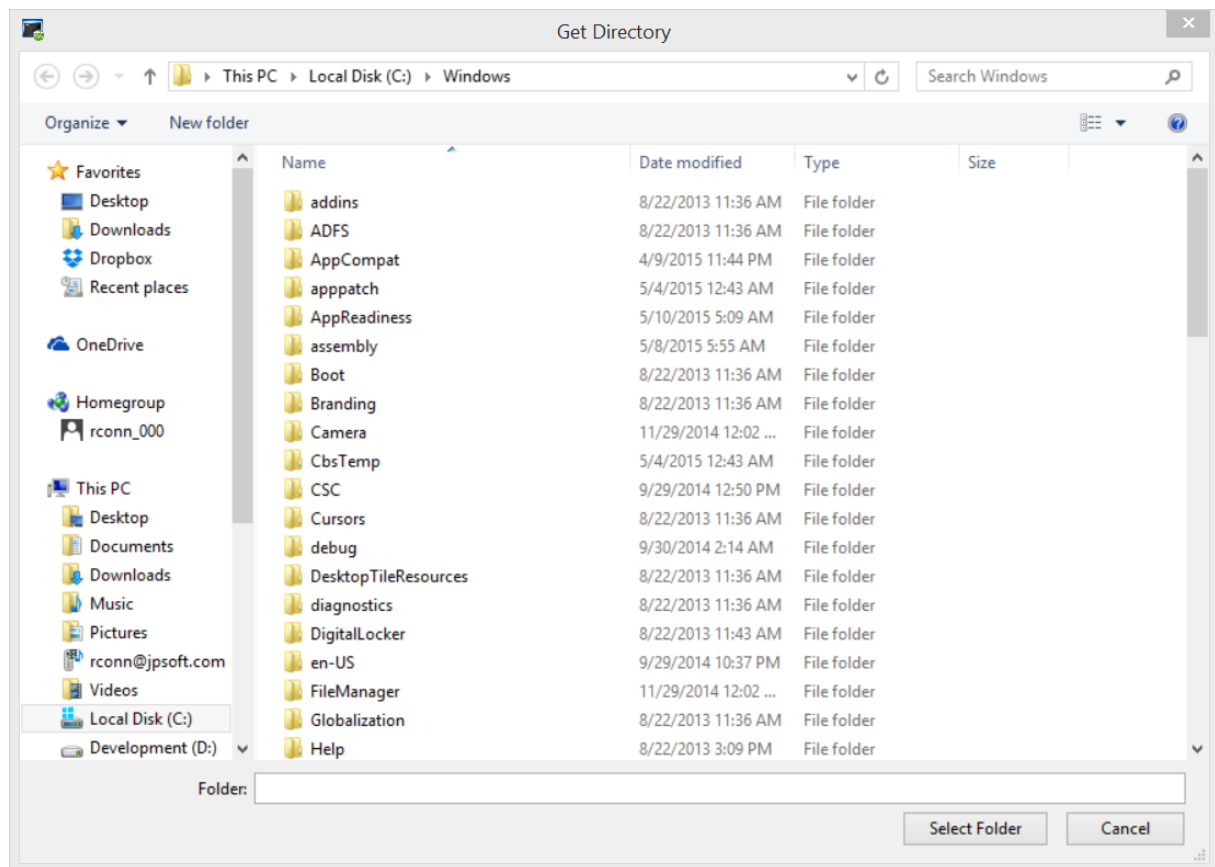

Note: @GETDIR deals with directories. All directories are folders, but not all folders are directories. To select a symbolic folder, see @GETFOLDER.

4.3.4.142 @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 for additional details.

If *filename* is quoted, the returned filename will also be quoted (if necessary).

@GETFILE accepts an optional third parameter to set the title of the dialog box.

If you're looking for directories, use @GETFOLDER.

Examples:

```
echo %@getfile[*]
```
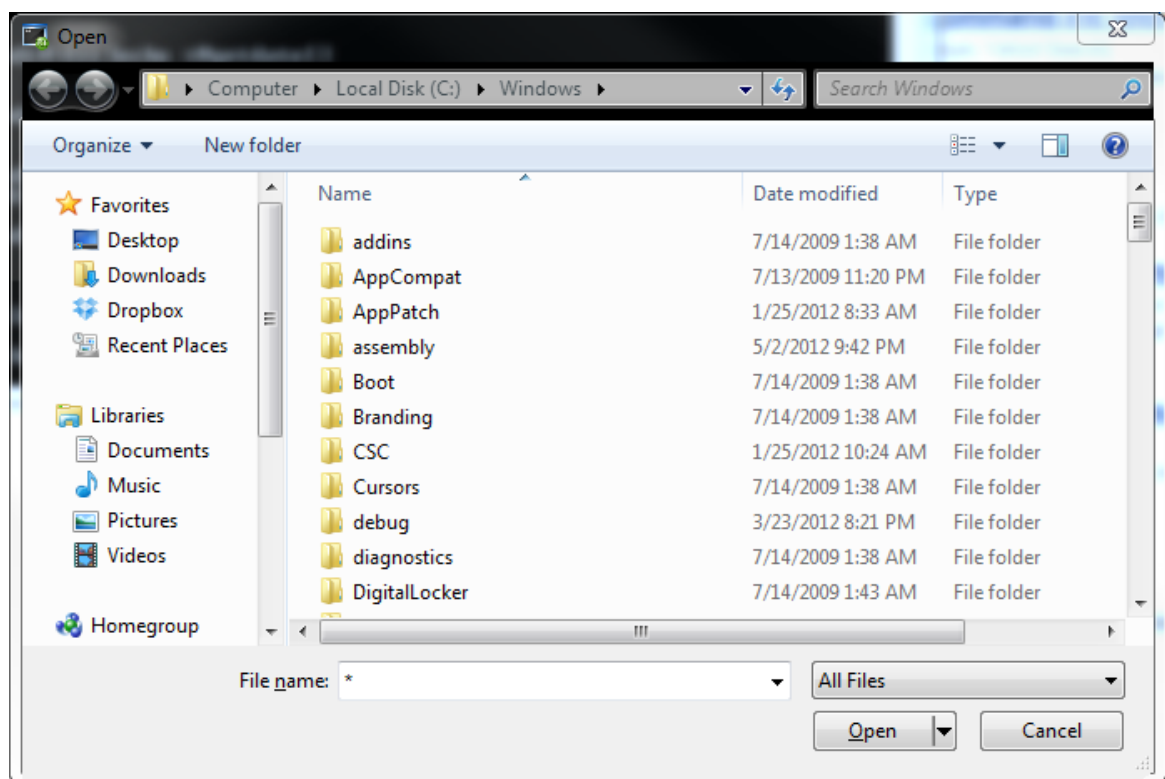


```
echo %@getfile["%windir",*.exe]
```
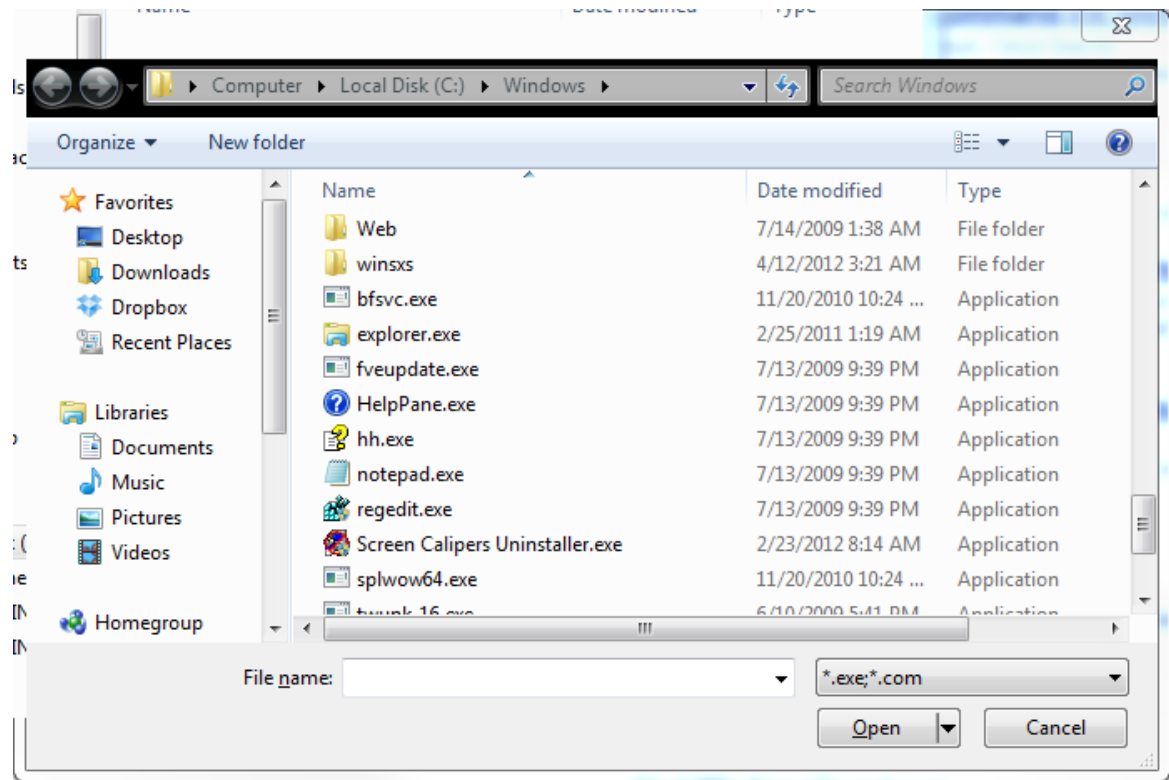
4.3.4.143 @GETFOLDER

@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.

If *startdir* is quoted, the returned folder name will also be quoted (if necessary).

Examples:

```
echo %@getfolder[]
```
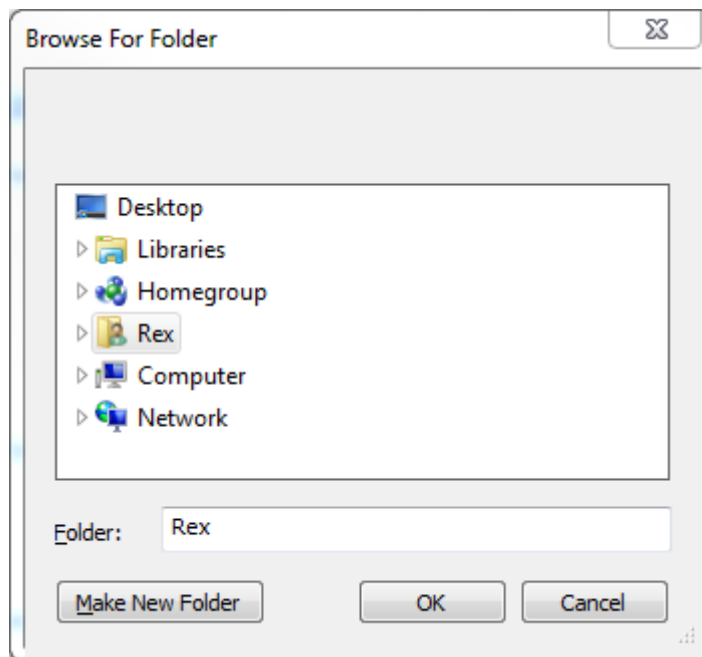
```
echo %@getfolder["c:\windows"]
```
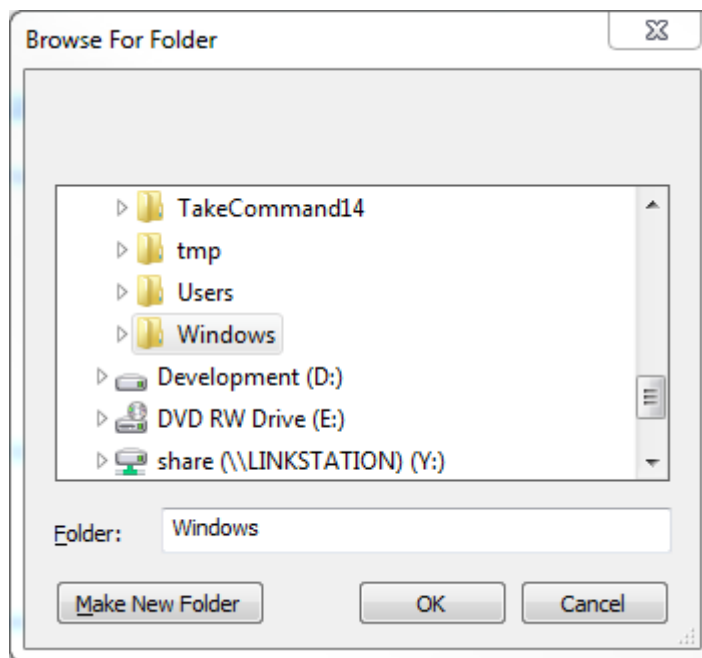

Note: @GETFOLDER deals with folders. All directories are folders, but not all folders are directories. To select a directory, see @GETDIR.

4.3.4.144 @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.

4.3.4.145 @HEXDECODE

@HEXDECODE[s,*string***]** : Create a text string from a hexadecimal input string. Returns the text string.
@HEXDECODE[*inputfile,outputfile***]** : Decode a hex encoded file. Returns 0 if the output file was successfully written.

Example:

```
echo %@hexdecode[s,656e636f64652074686869732073747472696e67]
encode this string
```

4.3.4.146 @HEXENCODE

@HEXENCODE[s,*string***]** : Create a hexadecimal string from a text input string. Returns the hex string.
@HEXENCODE[*inputfile,outputfile***]** : Encode a text file as a hex encoded file. Returns 0 if the output file was successfully written.

Example:

echo %@hexencode[s,encode this string]
656e636f64652074686869732073747472696e67

4.3.4.147 @HTMLDECODE

@HTMLDECODE[*string***]** : Decode an HTML string. The HTML escaped characters (i.e., **>**) are replaced with their original values.

Example:

```
echo "%@htmldecode[This is &amp; a string]"
"This is & a string"
```

See also TPIPE.

4.3.4.148 @HTMLENCODE

@HTMLENCODE[*string***]** : Encode a string for HTML, replacing characters like **> < &** with the HTML escaped characters (i.e., **>** for **>**).

Example:

```
echo "%@htmlencode[This is & a string]"
"This is &amp; a string"
```

See also TPIPE.

4.3.4.149 @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 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]
Tue
```

```
echo %@idow[%_date]
Sun
```

See also: @DOW.

4.3.4.150 @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 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[01-01-2010]
Tuesday
```

```
echo %@idowf[%_date]
Sunday
```

See also: @DOWF.

4.3.4.151 @IF

@IF[*condition,string1,string2***]**: Evaluates **condition** according to the rules described in Conditional Expressions, 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
```

4.3.4.152 @INC

@INC[*string***]** returns

- **1** if *string* is empty
- otherwise the same value as @EVAL[*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]
```

Example:

```
set start=5
set result=%@inc[start]
echo %result
6
```

4.3.4.153 @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 in **string2** to escape the comma
        ```
        echo %@index["TCC, Take Command, TCCLE",^,,2]
        ```

- searching for a **double quote** :

    1. use escape character 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 for details.

Examples:

In all examples below

- **string1**:   `This is a fine help file`
- **string2:   h**

n	result	purpose
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

4.3.4.154 @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\tcmd.ini,TakeCommand,history]
```

```
returns the size of the command history if it is specified in TCMD.INI.
```

4.3.4.155 @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 **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.

If **entry** is empty, the matching **section** (if any) is deleted.

Examples

    echo %@iniwrite[c:\tcmd\tcmd.ini,TakeCommand,history,8192]

will set the size of the command history to 8,192 bytes.

    echo %@iniwrite[c:\tcmd\tcmd.ini,TakeCommand,history,]

will remove the **history** entry from the file.

4.3.4.156 @INODE

@INODE[*filename***]** : Returns the inode (in hex) for the specified file.

When files are hard-linked to one another (see MKLNK), they share the same inode.

@INODE may not work for remote files (depending on your network redirector and the type of server you are querying).

Example:

    echo %@inode[c:\windows\explorer.exe]

```
00040000:000199D3
```

4.3.4.157 @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 form the end of **string2** (but not past its beginning). Setting **offset** to **-0** is the same as setting it to **0**, i.e., **string1** will precede **string2** in the result. To include a comma in **string1**, precede it with your escape character.

Numeric input may be entered in either decimal format (a sequence of 0-9 digits) 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

4.3.4.158 @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 is an older version of the same function.

Numeric input may be entered in either decimal format (a sequence of 0-9 digits) 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=

4.3.4.159 @INT

@INT[*n***]**: Returns the integer part of the number **n**. @INT will perform an implicit @EVAL on its argument, so you can use an arithmetic expression for **n.**

Examples:

```
echo %@int[1234]
1234

echo %@int[1.234]
1

echo %@int[12.34]
12
```

4.3.4.160 @IPADDRESS

@IPADDRESS[hostname]: Returns the numeric IP address for the specified hostname. The result is displayed in the standard format nnn.nnn.nnn.nnn. An invalid or unknown hostname will return an error (see @ERRTEXT to decipher the error number if necessary).

See also @IPNAME.

Example:

```
echo %@ipaddress[jpsoft.com]

141.101.124.120
```

4.3.4.161 @IPADDRESSN

@IPADDRESSN[*n***]**: Returns the IP address of the adapter at index *n*

Example:

```
echo %@ipaddressn[0]
192.168.1.25
```

4.3.4.162 @IPALIASES

@IPALIASES[*name***]**: Returns the other names (if any) corresponding to the host with the specified name.

4.3.4.163 @IPBROADCAST

@IPBROADCAST[*adapter***] :** The broadcast address of the specified network adapter.

adapter - The index of the adapter

4.3.4.164 @IPDESC

@IPDESC[*n***]**: Returns the description for the adapter at index *n*

Example:

```
echo %@ipdesc[0]
```

```
NETGEAR WNA3100 N300 Wireless USB Adapter
```

4.3.4.165 @IPDHCP

@IPDHCP[*n*]: Returns the DHCP server for the adapter at index *n*

Example:

```
echo %@ipdhcp[0]
192.168.1.254
```

4.3.4.166 @IPDHCPENABLED

@IPDHCPENABLED[*adapter*] - Returns 1 if the specified network adapter has DHCP enabled

adapter - The index of the adapter

4.3.4.167 @IPEXPIRES

@IPEXPIRES[*adapter*] : The expiration date and time of the lease obtained by the specified network adapter.

adapter - the index of the adapter

4.3.4.168 @IPGATEWAY

@IPGATEWAY[*n*]: Returns the gateway for the adapter at index *n*

Example:

```
echo %@ipgateway[0]
192.168.1.1
```


4.3.4.169 @IPIPV6LL

@IPIPV6LL[*adapter*] : The IPv6 link local address of the specified network adapter.

 adapter - the index of the adapter

4.3.4.170 @IPIPV6N

@IPIPV6N[*n*]: Returns the IPv6 address of the adapter at index *n*

Example:

```
echo %@ipipv6n[0]
fe80::e432:b8ae:c538:4191
```


4.3.4.171 @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 to decipher the error number if necessary).

See also: @IPADDRESS.

Examples:

```
echo %@ipname[173.194.43.40]
lga15s35-in-f8.1e100.net

echo %@ipname[0]
ASUS-PC
```

4.3.4.172 @IPNAMEN

@IPNAMEN[*n*]: Returns the name of the adapter at index *n*

Example:

```
echo %@ipnamen[0]
{E80FB8C8-0218-3B34-1188-528293717098}
```

4.3.4.173 @IPOBTAINED

@IPOBTAINED[*adapter*] : The date and time of when the current lease was obtained by the network adapter.

adapter - the index of the adapter

4.3.4.174 @IPOTHER

@IPOTHER[name, address] : Returns a space-delimited list of alternate addresses for the specified host (if any).

name - the host name
address - the host address

Most hosts have only one IP interface; this function is for querying multihomed hosts (hosts with more than one interface).

4.3.4.175 @IPOTHERL

@IPOTHERL[*adapter*] : Returns a space-delimited list of any other IP addresses leased by the specified network adapter.

adapter - the index of the adapter

4.3.4.176 @IPPHYSICAL

@IPPHYSICAL[*n*]: Returns the physical address of the adapter at index *n*

Example:

```
echo %@ipphysical[0]
30-4a-92-3e-66-1b
```

4.3.4.177 @IPPORT

@IPPORT[service**]** : Returns the port number for the specified service.

4.3.4.178 @IPSERVICEALIASES

@IPSERVICEALIASES[service**]** : Returns the aliases for the specified service.

4.3.4.179 @IPSTATUS

@IPSTATUS[adapter**]** : Returns the current status of the specified network adapter.

adapter - the index of the adapter

Possible return values are:

    Up
    Down
    Testing
    Unknown
    Dormant
    NotPresent
    LowerLayerDown

Examples:

```
echo %@ipstatus[0]
down

echo %@ipstatus[3]
up
```

4.3.4.180 @IPSUBNET

@IPSUBNET[n**]** : Returns the subnet mask of the adapter at index *n*.

Example:

```
 echo %@ipsubnet[0]
 255.255.255.0
```

4.3.4.181 @IPTYPE

@IPTYPE[n**]** : Returns the type of the adapter at index *n*. Possible values include:

    OTHER
    WIRELESS
    ETHERNET
    TOKENRING
    FDDI
    PPP
    LOOPBACK

SLIP

Example:

```
echo %@iptype[0]
WIRELESS

echo %@iptype[1]
ETHERNET
```

4.3.4.182 @IPWINS

@IPWINS[*n*] : Returns 1 if the adapter at index *n* uses WINS.

Example:

```
echo %@ipwins[0]
0
```

4.3.4.183 @IPWINSSERVER

@IPWINS[*n*] : Returns 1 if the adapter at index *n* uses WINS.

Example:

```
echo %@ipwins[0]
0
```

4.3.4.184 @IPWINSSERVER2

@IPWINSSERVER2[*adapter*] : Returns the secondary WINS server for the specified network adapter.

adapter - the index of the adapter

4.3.4.185 @IPZONEID

@IPZONEID[*n*] : Returns the IPv6 Zone ID (also known as a scope ID) for the adapter at index *n*. The values of the Zone ID are defined relative to the sending host

Example:

```
echo %@ipzoneid[0]
```

4.3.4.186 @ISALNUM

@ISALPHA[*string*]: Returns **1** if **string** is entirely composed of alphabetic (a-z, A-Z) characters; **0** otherwise.

See also: @ISALNUM, @ISASCII, @ISCNTRL, @ISDIGIT, @ISPRINT, @ISPUNCT, @ISSPACE, @ISXDIGIT.

Examples:

```
echo %@isalpha[abc]
1

echo %@isalpha[ABC]
1

echo %@isalpha[A B C]
0
```

4.3.4.187 @ISALPHA

@ISALPHA[*string***]**: Returns **1** if *string* is entirely composed of alphabetic (a-z, A-Z) characters; **0** otherwise.

See also: @ISALNUM, @ISASCII, @ISCNTRL, @ISDIGIT, @ISPRINT, @ISPUNCT, @ISSPACE, @ISXDIGIT.

Example:

```
echo %@isalpha[abc]
1

echo %@isalpha[ABC]
1

echo %@isalpha[A B C]
0
```

4.3.4.188 @ISASCII

@ISASCII[*string***]**: Returns **1** if *string* is entirely composed of 7-bit ASCII characters (0x00 - 0x7F); **0** otherwise.

See also: @ISALNUM, @ISALPHA, @ISCNTRL, @ISDIGIT, @ISPRINT, @ISPUNCT, @ISSPACE, @ISXDIGIT.

Examples:

```
echo %@isascii[abc]
1

echo %@isascii[abc 123]
1

echo %@isascii["abc"a]
0
```

4.3.4.189 @ISCNTRL

@ISCNTRL[_string_**]**: Returns **1** if _string_ is entirely composed of ASCII control characters (0x00 - 0x1F or 0x7F); **0** otherwise.

See also: @ISALNUM, @ISALPHA, @ISASCII, @ISDIGIT, @ISPRINT, @ISPUNCT, @ISSPACE, @ISXDIGIT.

Examples:

```
echo %@iscntrl[abc]
0

set var=^r^n
echo %@iscntrl[%var]
1
```

4.3.4.190 @ISDIGIT

@ISDIGIT[_string_**]** : Returns **1** if _string_ is entirely composed of decimal digits (0- 9); **0** otherwise.

See also: @ISALNUM, @ISALPHA, @ISASCII, @ISCNTRL, @ISPRINT, @ISPUNCT, @ISSPACE, @ISXDIGIT.

Examples:

```
echo %@isdigit[0]
1

echo %@isdigit[123.456]
0

echo %@isdigit[-123]
0
```

4.3.4.191 @ISFLOAT

@ISFLOAT[_string_] : Returns 1 if _string_ is composed only of numeric characters, a decimal separator, and an optional sign and/or thousands separator(s). The decimal separator and thousands separator are determined by your default country settings.

Examples:

```
echo %@isfloat[1234]
0

echo %@isfloat[1234.5]
1
```

4.3.4.192 @ISLOWER

@ISLOWER[*string*] - returns 1 if *string* is composed only of lower case letters.

Examples:

```
echo %@islower[hello]
1

echo %@islower[Hello]
0
```

4.3.4.193 @ISODOWI

@ISODOWI[*date*] : Returns the ISO 8601 numeric day of the week (Monday=1, Sunday=7).

Examples:

```
echo %@isodowi[%_date]
7

echo %@isodowi[2012-01-02]
1
```

4.3.4.194 @ISOWEEK

@ISOWEEK[*date*] : Returns the ISO8601 numeric week of year.

Examples:

```
echo %@isoweek[%_date]
23

echo %@isoweek[2012-23-02]
1
```

4.3.4.195 @ISOWYEAR

@ISOWYEAR[*date*] : Returns the ISO8601 numeric week date year.

Example:

```
echo %@isowyear[%_date]
2021
```

4.3.4.196 @ISPRIME

@ISPRIME[*n*] : Returns 1 if the (64-bit integer) *n* is a prime number.

Examples:

```
echo %@isprime[7]
1

echo %@isprime[22]
```

```
0

echo %@isprime[30181]
1
```

4.3.4.197 @ISPRINT

@ISPRINT[_string_**]**: Returns **1** if _string_ is entirely composed of printable characters; **0** otherwise.

See also: @ISALNUM, @ISALPHA, @ISASCII, @ISCNTRL, @ISDIGIT, @ISPUNCT, @ISSPACE, @ISXDIGIT.

Examples:

```
echo %@isprint[abc]
1

set var=abc^ndef
echo %@isprint[%var]
0
```

4.3.4.198 @ISPROC

@ISPROC[_pid_**]** : Returns 1 if the specified process ID is an active process, or 0 if it is not.

Examples:

```
echo %@pid[tcc.exe]
447988

echo %@isproc[447988]
1
```

4.3.4.199 @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, @ISALPHA, @ISASCII, @ISCNTRL, @ISDIGIT, @ISPRINT, @ISSPACE, @ISXDIGIT.

Examples:

```
echo %@ispunct[.]
1

echo %@ispunct[+]
1

echo %@ispunct[:-)]
1
```

```
echo %@ispunct[.,a]
0
```

4.3.4.200 @ISSPACE

@ISSPACE[_string_**]**: Returns **1** if _string_ is entirely composed of white space characters (0x09 - 0x0D or 0x20); **0** otherwise.

See also: @ISALNUM, @ISALPHA, @ISASCII, @ISCNTRL, @ISDIGIT, @ISPRINT, @ISPUNCT, @ISXDIGIT.

Example:

```
echo %@isspace[ ]
1

echo %@isspace[hello world]
0
```

4.3.4.201 @ISUPPER

@ISUPPER[_string_] : Returns 1 if _string_ is composed only of upper case letters.

Example:

```
echo %@isupper[HELLO]
1

echo %@isupper[Hello]
0
```

4.3.4.202 @ISXDIGIT

@ISXDIGIT[_string_**]**: Returns **1** if _string_ is entirely composed of hexadecimal digits (0- 9,A-F, a-f); **0** otherwise.

See also: @ISALNUM, @ISALPHA, @ISASCII, @ISCNTRL, @ISDIGIT, @ISPRINT, @ISPUNCT, @ISSPACE.

Example:

```
echo %@isxdigit[0]
0

echo %@isxdigit[7F]
1

echo %@isxdigit[0x7F]
1
```

4.3.4.203 @JSONCLOSE

@JSONCLOSE[] : Close a JSON file opened by @JSONOPEN.

Returns 0 on success, or a JSON error code on failure.

Example:

This batch file creates a JSON file named *d:\fido.json*, writes the opening brace, writes a property, writes the closing brace, and exits:

```
echo %@jsoncreate[d:\fido.json]
echo %@jsonStartObject[]
echo %@jsonPutProperty["name", "fido", 2]
echo %@jsonEndObject[]
echo %@jsonFlush[]
echo %@jsonclose[]
```

The resulting file *d:\json* looks like this:

```
{"name":"fido"}
```

JSON Errors

10231   Unbalanced element tag

10232   Invalid JSON markup

10233   Invalid XPath

10234   DOM tree unavailable

XMLp Errors

101   Invalid attribute index

102   No attributes available

103   Invalid namespace index

104   No namespaces available

105   Invalid element index

106   No elements available

107   Attribute does not exist

201   Unbalanced element tag

202   Unknown element prefix (can't find namespace)

203   Unknown attribute prefix (can't find namespace)

204   Invalid XML markup

205   Invalid end state for parser

206   Document contains unbalanced element

207   Invalid XPath

208   No such child

209   Top element does not match start of path

210   DOM tree unavailable

302 Can't open file.

4.3.4.204 @JSONCREATE

@JSONCREATE[*filename*] : Create a JSON file for writing by other JSON variable functions (for example, @JSONSTARTOBJECT, @JSONPUTPROPERTY, etc.).

If a JSON file is already open it will be closed before the new file is created. If the file already exists, @JSONCREATE will return an error.

Returns 0 on success, or a JSON error code on failure.

Example:

This batch file creates a JSON file named *d:\fido.json*, writes the opening brace, writes a property, writes the closing brace, and exits:

```
echo %@jsoncreate[d:\fido.json]
echo %@jsonStartObject[]
echo %@jsonPutProperty["name", "fido", 2]
echo %@jsonEndObject[]
echo %@jsonFlush[]
echo %@jsonclose[]
```

The resulting file *d:\json* looks like this:

```
{"name":"fido"}
```

JSON Errors

10231 Unbalanced element tag

10232 Invalid JSON markup

10233 Invalid XPath

10234 DOM tree unavailable

XMLp Errors

101 Invalid attribute index

102 No attributes available

103 Invalid namespace index

104 No namespaces available

105 Invalid element index

106 No elements available

107 Attribute does not exist

201 Unbalanced element tag

202 Unknown element prefix (can't find namespace)

203 Unknown attribute prefix (can't find namespace)

204 Invalid XML markup

205 Invalid end state for parser

206   Document contains unbalanced element

207   Invalid XPath

208   No such child

209   Top element does not match start of path

210   DOM tree unavailable

302   Can't open file.

4.3.4.205 @JSONENDARRAY

@JSONENDARRAY[] : Write the closing bracket of a JSON array.

Returns 0 on success, or a JSON error code on failure.

See also @JSONSTARTARRAY.

Example:

This batch file creates a JSON file named *d:\fido.json*, writes the opening brace, writes a property, creates an array, writes two values into the array, closes the array, writes the closing brace, and closes the file:

```
echo %@jsoncreate[d:\fido.json]
echo %@jsonStartObject[]
echo %@jsonPutProperty["name", "fido", 2]
echo %@jsonPutName["previousOwners"]
echo %@jsonStartArray[]
echo %@jsonPutValue["Steve Widgetson",2]
echo %@jsonPutValue["Wanda Widgetson", 2]
echo %@jsonEndarray[]
echo %@jsonEndObject[]
echo %@jsonFlush[]
echo %@jsonclose[]
```

The resulting file *d:\json* looks like this:

```
{"name":"fido","previousOwners":["Steve Widgetson","Wanda Widgetson"]}
```

JSON Errors

10231   Unbalanced element tag

10232   Invalid JSON markup

10233   Invalid XPath

10234   DOM tree unavailable

XMLp Errors

101   Invalid attribute index

102   No attributes available

103   Invalid namespace index

104   No namespaces available

105   Invalid element index

106   No elements available

107   Attribute does not exist

201   Unbalanced element tag

202   Unknown element prefix (can't find namespace)

203   Unknown attribute prefix (can't find namespace)

204   Invalid XML markup

205   Invalid end state for parser

206   Document contains unbalanced element

207   Invalid XPath

208   No such child

209   Top element does not match start of path

210   DOM tree unavailable

302   Can't open file.

4.3.4.206 @JSONENDOBJECT

@JSONENDOBJECT[] : Write the closing brace of a JSON object.

Returns 0 on success, or a JSON error code on failure.

See also @JSONSTARTOBJECT.

Example:

This batch file creates a JSON file named *d:\fido.json*, writes the opening brace, writes a property, writes the closing brace, and closes the file:

```
echo %@jsoncreate[d:\fido.json]
echo %@jsonStartObject[]
echo %@jsonPutProperty["name", "fido", 2]
echo %@jsonEndObject[]
echo %@jsonFlush[]
echo %@jsonclose[]
```

The resulting file *d:\json* looks like this:

```
{"name":"fido"}
```

JSON Errors

10231   Unbalanced element tag

10232   Invalid JSON markup

10233   Invalid XPath

10234   DOM tree unavailable

XMLp Errors

101 Invalid attribute index

102 No attributes available

103 Invalid namespace index

104 No namespaces available

105 Invalid element index

106 No elements available

107 Attribute does not exist

201 Unbalanced element tag

202 Unknown element prefix (can't find namespace)

203 Unknown attribute prefix (can't find namespace)

204 Invalid XML markup

205 Invalid end state for parser

206 Document contains unbalanced element

207 Invalid XPath

208 No such child

209 Top element does not match start of path

210 DOM tree unavailable

302 Can't open file.

4.3.4.207 @JSONFLUSH

@JSONFLUSH[] : Flush the JSON parser buffers.

@JSONFLUSH will write the current JSON buffer to disk if it has changed.

Example:

This batch file creates a JSON file named *d:\fido.json*, writes the opening brace, writes a property, writes the closing brace, and closes the file:

```
echo %@jsoncreate[d:\fido.json]
echo %@jsonStartObject[]
echo %@jsonPutProperty["name", "fido", 2]
echo %@jsonEndObject[]
echo %@jsonFlush[]
echo %@jsonclose[]
```

The resulting file *d:\json* looks like this:

```
{"name":"fido"}
```

JSON Errors

10231 Unbalanced element tag

10232 Invalid JSON markup

10233 Invalid XPath

10234   DOM tree unavailable

XMLp Errors

101   Invalid attribute index

102   No attributes available

103   Invalid namespace index

104   No namespaces available

105   Invalid element index

106   No elements available

107   Attribute does not exist

201   Unbalanced element tag

202   Unknown element prefix (can't find namespace)

203   Unknown attribute prefix (can't find namespace)

204   Invalid XML markup

205   Invalid end state for parser

206   Document contains unbalanced element

207   Invalid XPath

208   No such child

209   Top element does not match start of path

210   DOM tree unavailable

302   Can't open file.

4.3.4.208 @JSONHASXPATH

@JSONHASXPATH[[*"filename",]xpath*]] : Returns 1 if the specified *xpath* exists in the JSON file, or 0 if it doesn't.

The *xpath* always begins with /json.

If you do not specify a filename, @JSONHASXPATH will use the file previously opened by @JSONOPEN.

Example:

For example, with this JSON file:

```
{
  "firstlevel": {
    "one": "value",
    "two": ["first", "second"],
    "three": "value three"
  }
}

echo %@jsonnasxpath["test.json",/json/firstlevel/one/]
1
```

JSON Errors

10231   Unbalanced element tag.

10232   Invalid JSON markup.

10233   Invalid XPath.

10234   DOM tree unavailable (set BuildDOM to true and reparse).

XMLp Errors

101   Invalid attribute index.

102   No attributes available.

103   Invalid namespace index.

104   No namespaces available.

105   Invalid element index.

106   No elements available.

107   Attribute does not exist.

201   Unbalanced element tag.

202   Unknown element prefix (can't find namespace).

203   Unknown attribute prefix (can't find namespace).

204   Invalid XML markup.

205   Invalid end state for parser.

206   Document contains unbalanced elements.

207   Invalid XPath.

208   No such child.

209   Top element does not match start of path.

210   DOM tree unavailable

302   Can't open file.

4.3.4.209 @JSONINPUT

@JSONINPUT[*inputdata*] : Parse an input string as JSON data. (Use this instead of @JSONOPEN if you don't have an input file.)

@JSONINPUT will parse the input string and create an internal JSON document. You can modify the document with the other @JSONxxx commands (such as @JSONINSERTVALUE) and then save the document to disk with @JSONSAVE and @JSONCLOSE.

Returns 0 on success, or a JSON error code on failure.

Example:

Pass a JSON string to @JSONINPUT, and write it to the file *d:\json* :

```
echo %@jsoninput[{"name":"fido"}]
echo %@jsonsave[d:\fido.json]
echo %@jsonclose[]
```

JSON Errors

10231 Unbalanced element tag

10232 Invalid JSON markup

10233 Invalid XPath

10234 DOM tree unavailable

XMLp Errors

101 Invalid attribute index

102 No attributes available

103 Invalid namespace index

104 No namespaces available

105 Invalid element index

106 No elements available

107 Attribute does not exist

201 Unbalanced element tag

202 Unknown element prefix (can't find namespace)

203 Unknown attribute prefix (can't find namespace)

204 Invalid XML markup

205 Invalid end state for parser

206 Document contains unbalanced element

207 Invalid XPath

208 No such child

209 Top element does not match start of path

210 DOM tree unavailable

302 Can't open file.

4.3.4.210 @JSONINSERTPROPERTY

@JSONINSERTPROPERTY[xpath,name,*value,type,position*] : Writes a value of a property.

Name specifies the name of the property.

Value specifies the new value.

Type specifies the type of the value. Possible values are:
- 0 (Object)
- 1 (Array)
- 2 (String)
- 3 (Number)
- 4 (Bool)
- 5 (Null)
- 6 (Raw)

The *Position* parameter specifies the position of Value relative to the element specified by *XPath*. Possible values are:
- 0 (Before the current element)

- 1 (After the current element)
- 2 (The first child of the current element)
- 3 (The last child of the current element)

The JSON file must have been opened with a previous call to @JSONOPEN.

Example:

If you have a JSON file like this:

```
{
  "store": {
    "books": [
      {
        "category": "reference",
        "author": "Nigel Rees",
        "title": "Sayings of the Century",
      },
      {
        "category": "fiction",
        "author": "Evelyn Waugh",
        "title": "Sword of Honour",
      }
    ]
  }
}
```

To insert a new property "price" for each book:

```
echo %@jsoninsertproperty[/json/store/books/[1],"price","8.95",3,3]
echo %@jsoninsertproperty[/json/store/books/[1],"price","12.99",3,3]
```

This will produce the JSON:

```
{
  "store": {
    "books": [
      {
        "category": "reference",
        "author": "Nigel Rees",
        "title": "Sayings of the Century",
        "price": 8.95
      },
      {
        "category": "fiction",
        "author": "Evelyn Waugh",
        "title": "Sword of Honour",
        "price": 12.99
```

```
          }
        ]
      }
    }
```

JSON Errors

10231   Unbalanced element tag

10232   Invalid JSON markup

10233   Invalid XPath

10234   DOM tree unavailable

XMLp Errors

101   Invalid attribute index

102   No attributes available

103   Invalid namespace index

104   No namespaces available

105   Invalid element index

106   No elements available

107   Attribute does not exist

201   Unbalanced element tag

202   Unknown element prefix (can't find namespace)

203   Unknown attribute prefix (can't find namespace)

204   Invalid XML markup

205   Invalid end state for parser

206   Document contains unbalanced element

207   Invalid XPath

208   No such child

209   Top element does not match start of path

210   DOM tree unavailable

302   Can't open file.

4.3.4.211 @JSONINSERTVALUE

@JSONINSERTVALUE[xpath,*value,type,position*] : Inserts the specified value at the selected position.

Value specifies the new value.

Type specifies the type of the value. Possible values are:
- 0 (Object)

- 1 (Array)
- 2 (String)
- 3 (Number)
- 4 (Bool)
- 5 (Null)
- 6 (Raw)

The *Position* parameter specifies the position of Value relative to the element specified by *XPath*. Possible values are:

- 0 (Before the current element)
- 1 (After the current element)
- 2 (The first child of the current element)
- 3 (The last child of the current element)

The JSON file must have been opened with a previous call to @JSONOPEN.

Returns 0 on success, or a JSON error code on failure.

For example, if you have a JSON file like this:

```
{
  "store": {
    "books": [
      {
        "category": "reference",
        "author": "Nigel Rees",
        "title": "Sayings of the Century",
      },
      {
        "category": "fiction",
        "author": "Evelyn Waugh",
        "title": "Sword of Honour",
      }
    ]
  }
}
```

To add a new book to the array:

```
echo %@jsoninsertvalue[/json/store/books,"",0,3]
echo %@jsoninsertproperty[/json/store/books/[3],"category","fiction",2,3]
echo %@jsoninsertproperty[/json/store/books/[3],"author","Herman
Melville",2,3]
echo %@jsoninsertproperty[/json/store/books/[3],"title","Moby Dick",2,3]
```

JSON Errors

10231   Unbalanced element tag

10232   Invalid JSON markup

10233   Invalid XPath

10234   DOM tree unavailable

XMLp Errors

101   Invalid attribute index

102   No attributes available

103   Invalid namespace index

104   No namespaces available

105   Invalid element index

106   No elements available

107   Attribute does not exist

201   Unbalanced element tag

202   Unknown element prefix (can't find namespace)

203   Unknown attribute prefix (can't find namespace)

204   Invalid XML markup

205   Invalid end state for parser

206   Document contains unbalanced element

207   Invalid XPath

208   No such child

209   Top element does not match start of path

210   DOM tree unavailable

302   Can't open file.

4.3.4.212 @JSONNODENAMES

@JSONNODENAMES["*filename",]xpath*] : Returns a space delimited list of the element names for the specified xpath..

If you do not specify a filename, @JSONNODES will use the file previously opened by @JSONOPEN.

Example:

For example, with this JSON file:

```
{
  "firstlevel": {
    "one": "value",
    "two": ["first", "second"],
    "three": "value three"
  }
}

echo %@jsonnodenames["test.json",/json/firstlevel/]
```

```
"one" "two" "three"
```

JSON Errors

10231   Unbalanced element tag

10232   Invalid JSON markup

10233   Invalid XPath

10234   DOM tree unavailable

XMLp Errors

101   Invalid attribute index

102   No attributes available

103   Invalid namespace index

104   No namespaces available

105   Invalid element index

106   No elements available

107   Attribute does not exist

201   Unbalanced element tag

202   Unknown element prefix (can't find namespace)

203   Unknown attribute prefix (can't find namespace)

204   Invalid XML markup

205   Invalid end state for parser

206   Document contains unbalanced element

207   Invalid XPath

208   No such child

209   Top element does not match start of path

210   DOM tree unavailable

302   Can't open file.

4.3.4.213 @JSONNODES

@JSONNODES[["*filename",]path*] : Returns the number of nodes (children) for the specified path in a JSON file.

If you do not specify a filename, @JSONNODES will use the file previously opened by @JSONOPEN.

Example:

For example, with this JSON file:

```
{
  "firstlevel": {
    "one": "value",
    "two": ["first", "second"],
    "three": "value three"
  }
}
```

```
echo %@jsonnodes["test.json",/json/firstlevel/]
"value"
```

JSON Errors

10231   Unbalanced element tag

10232   Invalid JSON markup

10233   Invalid XPath

10234   DOM tree unavailable

XMLp Errors

101   Invalid attribute index

102   No attributes available

103   Invalid namespace index

104   No namespaces available

105   Invalid element index

106   No elements available

107   Attribute does not exist

201   Unbalanced element tag

202   Unknown element prefix (can't find namespace)

203   Unknown attribute prefix (can't find namespace)

204   Invalid XML markup

205   Invalid end state for parser

206   Document contains unbalanced element

207   Invalid XPath

208   No such child

209   Top element does not match start of path

210   DOM tree unavailable

302   Can't open file.

4.3.4.214 @JSONOPEN

@JSONOPEN[*filename*] **:** Open a JSON file for use by other JSON variable functions (for example, @JSONXPATH, @JSONNODES, @JSONINPUTVALUE, etc.).

Returns 0 on success, or a JSON error code on failure.

Example:

For example, with this JSON file *level.json* :

```
{
  "firstlevel": {
    "one": "value",
    "two": ["first", "second"],
    "three": "value three"
  }
```

```
}

echo %@jsonopen[level.json]
echo %@jsonxpath[/json/firstlevel/one/]
"value"

echo %@jsonxpath[/json/firstlevel/two/[2]/]
"second"
```

JSON Errors

10231   Unbalanced element tag

10232   Invalid JSON markup

10233   Invalid XPath

10234   DOM tree unavailable

XMLp Errors

101   Invalid attribute index

102   No attributes available

103   Invalid namespace index

104   No namespaces available

105   Invalid element index

106   No elements available

107   Attribute does not exist

201   Unbalanced element tag

202   Unknown element prefix (can't find namespace)

203   Unknown attribute prefix (can't find namespace)

204   Invalid XML markup

205   Invalid end state for parser

206   Document contains unbalanced element

207   Invalid XPath

208   No such child

209   Top element does not match start of path

210   DOM tree unavailable

302   Can't open file.

4.3.4.215 @JSONOUTPUT

@JSONOUTPUT[*data*] : Output JSON to a string after processing. (Use this instead of @JSONSAVE if you don't want to create a file.)

Returns 0 on success, or a JSON error code on failure.

JSON Errors

10231   Unbalanced element tag

10232   Invalid JSON markup

10233   Invalid XPath

10234   DOM tree unavailable

XMLp Errors

101   Invalid attribute index

102   No attributes available

103   Invalid namespace index

104   No namespaces available

105   Invalid element index

106   No elements available

107   Attribute does not exist

201   Unbalanced element tag

202   Unknown element prefix (can't find namespace)

203   Unknown attribute prefix (can't find namespace)

204   Invalid XML markup

205   Invalid end state for parser

206   Document contains unbalanced element

207   Invalid XPath

208   No such child

209   Top element does not match start of path

210   DOM tree unavailable

302   Can't open file.

4.3.4.216 @JSONPUTNAME

@JSONPUTNAME[*name*] : Writes the name of a property to a JSON file.

The file must have been opened with a previous @JSONOPEN.

Returns 0 on success, or a JSON error code on failure.

JSON Errors

10231   Unbalanced element tag

10232   Invalid JSON markup

10233   Invalid XPath

10234   DOM tree unavailable

XMLp Errors

101   Invalid attribute index

102   No attributes available

103   Invalid namespace index

104   No namespaces available

105   Invalid element index

106   No elements available

107 Attribute does not exist

201 Unbalanced element tag

202 Unknown element prefix (can't find namespace)

203 Unknown attribute prefix (can't find namespace)

204 Invalid XML markup

205 Invalid end state for parser

206 Document contains unbalanced element

207 Invalid XPath

208 No such child

209 Top element does not match start of path

210 DOM tree unavailable

302 Can't open file.

4.3.4.217 @JSONPUTPROPERTY

@JSONPUTPROPERTY[*name,value,type*] : Writes the name of a property and its value to a JSON file.

The file must have been opened with a previous @JSONOPEN.

The *name* parameter specifies the name of the property.

The *value* parameter specifies the value of the property.

The *type* parameter specifies the type of the value. Possible values are:
- 0 (Object)
- 1 (Array)
- 2 (String)
- 3 (Number)
- 4 (Bool)
- 5 (Null)
- 6 (Raw)

Returns 0 on success, or a JSON error code on failure.

Example:

This batch file creates a JSON file named *d:\fido.json*, writes the opening brace, writes a property, writes the closing brace, and closes the file:

```
echo %@jsoncreate[d:\fido.json]
echo %@jsonStartObject[]
echo %@jsonPutProperty["name", "fido", 2]
echo %@jsonEndObject[]
echo %@jsonFlush[]
echo %@jsonclose[]
```

The resulting file *d:\json* looks like this:

```
{"name":"fido"}
```

JSON Errors

10231   Unbalanced element tag

10232   Invalid JSON markup

10233   Invalid XPath

10234   DOM tree unavailable

XMLp Errors

101   Invalid attribute index

102   No attributes available

103   Invalid namespace index

104   No namespaces available

105   Invalid element index

106   No elements available

107   Attribute does not exist

201   Unbalanced element tag

202   Unknown element prefix (can't find namespace)

203   Unknown attribute prefix (can't find namespace)

204   Invalid XML markup

205   Invalid end state for parser

206   Document contains unbalanced element

207   Invalid XPath

208   No such child

209   Top element does not match start of path

210   DOM tree unavailable

302   Can't open file.

4.3.4.218 @JSONPUTRAW

@JSONPUTRAW[*text*] : Writes a raw JSON fragment to a JSON file.

The file must have been opened with a previous @JSONOPEN.

Returns 0 on success, or a JSON error code on failure.

JSON Errors

10231   Unbalanced element tag

10232   Invalid JSON markup

10233   Invalid XPath

10234   DOM tree unavailable

XMLp Errors

101   Invalid attribute index

102   No attributes available

103   Invalid namespace index

104   No namespaces available

105   Invalid element index

106   No elements available

107   Attribute does not exist

201   Unbalanced element tag

202   Unknown element prefix (can't find namespace)

203   Unknown attribute prefix (can't find namespace)

204   Invalid XML markup

205   Invalid end state for parser

206   Document contains unbalanced element

207   Invalid XPath

208   No such child

209   Top element does not match start of path

210   DOM tree unavailable

302   Can't open file.

4.3.4.219 @JSONPUTVALUE

@JSONPUTVALUE[*value,type*] : Writes the value of a property to a JSON file.

Value specifies the new value.

ValueType specifies the type of the value. Possible values are:
- 0 (Object)
- 1 (Array)
- 2 (String)
- 3 (Number)
- 4 (Bool)
- 5 (Null)
- 6 (Raw)

The file must have been opened with a previous @JSONOPEN.

Returns 0 on success, or a JSON error code on failure.

Example:

This batch file creates a JSON file named *d:\fido.json*, writes the opening brace, writes a property, creates an array, writes two values into the array, closes the array, writes the closing brace, and closes the file:

```
echo %@jsoncreate[d:\fido.json]
echo %@jsonStartObject[]
echo %@jsonPutProperty["name", "fido", 2]
echo %@jsonPutName["previousOwners"]
echo %@jsonStartArray[]
echo %@jsonPutValue["Steve Widgetson",2]
echo %@jsonPutValue["Wanda Widgetson", 2]
echo %@jsonEndarray[]
echo %@jsonEndObject[]
echo %@jsonFlush[]
echo %@jsonclose[]
```

The resulting file *d:\json* looks like this:

```
{"name":"fido","previousOwners":["Steve Widgetson","Wanda Widgetson"]}
```

JSON Errors

10231   Unbalanced element tag

10232   Invalid JSON markup

10233   Invalid XPath

10234   DOM tree unavailable

XMLp Errors

101   Invalid attribute index

102   No attributes available

103   Invalid namespace index

104   No namespaces available

105   Invalid element index

106   No elements available

107   Attribute does not exist

201   Unbalanced element tag

202   Unknown element prefix (can't find namespace)

203   Unknown attribute prefix (can't find namespace)

204   Invalid XML markup

205   Invalid end state for parser

206   Document contains unbalanced element

207   Invalid XPath

208   No such child

209   Top element does not match start of path

210   DOM tree unavailable

302   Can't open file.

4.3.4.220 @JSONREMOVE

@JSONREMOVE[*xpath*] : Removes the element or value set in XPath. The file must have been opened with a previous @JSONOPEN.

If *xpath* is not specified, @JSONREMOVE will use the current XPath.

Returns 0 on success, or a JSON error code on failure.

Example:

With this JSON file:

```
{
  "store": {
    "books": [
      {
        "category": "reference",
        "author": "Nigel Rees",
        "title": "Sayings of the Century",
      },
      {
        "category": "fiction",
        "author": "Evelyn Waugh",
        "title": "Sword of Honour",
      }
    ]
  }
}
```

To remove the "category" properties from each book:

```
%@jsonremove[/json/store/books/[1]/category]
%@jsonremove[/json/store/books/[2]/category]
```

To remove the first book:

```
%@jsonremove[/json/store/books/[1]]
```

JSON Errors

10231   Unbalanced element tag
10232   Invalid JSON markup
10233   Invalid XPath
10234   DOM tree unavailable

XMLp Errors

101   Invalid attribute index
102   No attributes available
103   Invalid namespace index

104   No namespaces available

105   Invalid element index

106   No elements available

107   Attribute does not exist

201   Unbalanced element tag

202   Unknown element prefix (can't find namespace)

203   Unknown attribute prefix (can't find namespace)

204   Invalid XML markup

205   Invalid end state for parser

206   Document contains unbalanced element

207   Invalid XPath

208   No such child

209   Top element does not match start of path

210   DOM tree unavailable

302   Can't open file.

4.3.4.221 @JSONRESET

@JSONRESET[] : Flush the JSON parser buffers, and initialize the parser to its default state.

Returns 0 on success, or a JSON error code on failure.

JSON Errors

10231   Unbalanced element tag

10232   Invalid JSON markup

10233   Invalid XPath

10234   DOM tree unavailable

XMLp Errors

101   Invalid attribute index

102   No attributes available

103   Invalid namespace index

104   No namespaces available

105   Invalid element index

106   No elements available

107   Attribute does not exist

201   Unbalanced element tag

202   Unknown element prefix (can't find namespace)

203   Unknown attribute prefix (can't find namespace)

204   Invalid XML markup

205   Invalid end state for parser

206   Document contains unbalanced element

207   Invalid XPath

208   No such child

209   Top element does not match start of path

210   DOM tree unavailable

302   Can't open file.

4.3.4.222 @JSONSAVE

@JSONSAVE[*outputfile[,overwrite]*] : Saves the modified JSON document to the specified output file.

If the optional *overwrite* argument is 1, @JSONSAVE will overwrite an existing file. Otherwise, @JSONSAVE will display an error.

@JSONSAVE would normally be used when you are creating a document using [@JSONINPUT](#).

@JSONSAVE returns 0 on success, or an error code on failure.

Example:

Pass a JSON string to @JSONINPUT, and write it to the file *d:\json* :

```
echo %@jsoninput[{"name":"fido"}]
echo %@jsonsave[d:\fido.json]
```

JSON Errors

10231   Unbalanced element tag

10232   Invalid JSON markup

10233   Invalid XPath

10234   DOM tree unavailable

XMLp Errors

101   Invalid attribute index

102   No attributes available

103   Invalid namespace index

104   No namespaces available

105   Invalid element index

106   No elements available

107   Attribute does not exist

201   Unbalanced element tag

202   Unknown element prefix (can't find namespace)

203   Unknown attribute prefix (can't find namespace)

204   Invalid XML markup

205   Invalid end state for parser

206   Document contains unbalanced element

207   Invalid XPath

208   No such child

209   Top element does not match start of path

210   DOM tree unavailable

302   Can't open file.

4.3.4.223 @JSONSETNAME

@JSONSETNAME[*[xpath,]name*] : Sets a new name for the element specified by XPath. The file must have been opened with a previous @JSONOPEN.

If *xpath* is not specified, @JSONSETNAME will default to the current *xpath*.

Returns 0 on success, or a JSON error code on failure.

Example:

With this JSON file:

```
{
  "store": {
    "books": [
      {
        "tags": ["trilogy", "war"],
        "category": "reference",
        "author": "Nigel Rees",
        "title": "Sayings of the Century",
      },
      {
        "tags": ["classic", "whales"],
        "category": "fiction",
        "author": "Evelyn Waugh",
        "title": "Sword of Honour",
      }
    ]
  }
}
```

To rename "tags" to meta:

```
%@jsonsetname[/json/store/books/[1]/tags,meta]
%@jsonsetname[/json/store/books/[2]/tags,meta]
```

JSON Errors

10231   Unbalanced element tag

10232   Invalid JSON markup

10233   Invalid XPath

10234   DOM tree unavailable

XMLp Errors

101   Invalid attribute index

102   No attributes available

103   Invalid namespace index
104   No namespaces available
105   Invalid element index
106   No elements available
107   Attribute does not exist
201   Unbalanced element tag
202   Unknown element prefix (can't find namespace)
203   Unknown attribute prefix (can't find namespace)
204   Invalid XML markup
205   Invalid end state for parser
206   Document contains unbalanced element
207   Invalid XPath
208   No such child
209   Top element does not match start of path
210   DOM tree unavailable
302   Can't open file.

4.3.4.224 @JSONSETVALUE

@JSONSETVALUE[[*xpath,*]*value,type*] **:** Sets a new value for the element specified by XPath. The file must have been opened with a previous @JSONOPEN.

If *xpath* is not specified, @JSONSETVALUE will default to the current *xpath***.**

value specifies the new value.

type specifies the type of the value. Possible values are:
- 0 (Object)
- 1 (Array)
- 2 (String)
- 3 (Number)
- 4 (Bool)
- 5 (Null)
- 6 (Raw)

Returns 0 on success, or a JSON error code on failure.

Example:

With this JSON file:

```
{
  "store": {
    "books": [
      {
        "tags": ["trilogy", "war"],
```

```
            "category": "reference",
            "author": "Nigel Rees",
            "title": "Sayings of the Century",
            "price": 12.99
        },
        {
            "tags": ["classic", "whales"],
            "category": "fiction",
            "author": "Evelyn Waugh",
            "title": "Sword of Honour",
            "price": 10.99
        }
    ]
  }
}
```

To update the price:

    %@jsonsetvalue[/json/store/books/[1]/price,13.99,3]

JSON Errors

10231   Unbalanced element tag

10232   Invalid JSON markup

10233   Invalid XPath

10234   DOM tree unavailable

XMLp Errors

101   Invalid attribute index

102   No attributes available

103   Invalid namespace index

104   No namespaces available

105   Invalid element index

106   No elements available

107   Attribute does not exist

201   Unbalanced element tag

202   Unknown element prefix (can't find namespace)

203   Unknown attribute prefix (can't find namespace)

204   Invalid XML markup

205   Invalid end state for parser

206   Document contains unbalanced element

207   Invalid XPath

208   No such child

209   Top element does not match start of path

210   DOM tree unavailable

302   Can't open file.

4.3.4.225 @JSONSTARTARRAY

@JSONSTARTARRAY[] : Writes the opening bracket of a JSON array. The file must have been opened with a previous @JSONOPEN.

Returns 0 on success, or a JSON error code on failure.

See also @JSONENDARRAY;

Example:

This batch file creates a JSON file named *d:\fido.json*, writes the opening brace, writes a property, creates an array, writes two values into the array, closes the array, writes the closing brace, and closes the file:

```
echo %@jsoncreate[d:\fido.json]
echo %@jsonStartObject[]
echo %@jsonPutProperty["name", "fido", 2]
echo %@jsonPutName["previousOwners"]
echo %@jsonStartArray[]
echo %@jsonPutValue["Steve Widgetson",2]
echo %@jsonPutValue["Wanda Widgetson", 2]
echo %@jsonEndarray[]
echo %@jsonEndObject[]
echo %@jsonFlush[]
echo %@jsonclose[]
```

The resulting file *d:\json* looks like this:

```
{"name":"fido","previousOwners":["Steve Widgetson","Wanda Widgetson"]}
```

JSON Errors

10231   Unbalanced element tag

10232   Invalid JSON markup

10233   Invalid XPath

10234   DOM tree unavailable

XMLp Errors

101   Invalid attribute index

102   No attributes available

103   Invalid namespace index

104   No namespaces available

105   Invalid element index

106   No elements available

107   Attribute does not exist

201   Unbalanced element tag

202   Unknown element prefix (can't find namespace)

203   Unknown attribute prefix (can't find namespace)

204   Invalid XML markup

205   Invalid end state for parser

206   Document contains unbalanced element

207   Invalid XPath

208   No such child

209   Top element does not match start of path

210   DOM tree unavailable

302   Can't open file.

4.3.4.226 @JSONSTARTOBJECT

@JSONSTARTOBJECT[] : Writes the opening brace of a JSON object. The file must have been opened with a previous @JSONOPEN.

Returns 0 on success, or a JSON error code on failure.

See also @JSONENDOBJECT.

Example:

This batch file creates a JSON file named *d:\fido.json*, writes the opening brace, writes a property, writes the closing brace, and closes the file:

```
echo %@jsoncreate[d:\fido.json]
echo %@jsonStartObject[]
echo %@jsonPutProperty["name", "fido", 2]
echo %@jsonEndObject[]
echo %@jsonFlush[]
echo %@jsonclose[]
```

The resulting file *d:\json* looks like this:

```
{"name":"fido"}
```

JSON Errors

10231   Unbalanced element tag

10232   Invalid JSON markup

10233   Invalid XPath

10234   DOM tree unavailable

XMLp Errors

101   Invalid attribute index

102   No attributes available

103   Invalid namespace index

104   No namespaces available

105   Invalid element index

106   No elements available

107   Attribute does not exist

201   Unbalanced element tag

202   Unknown element prefix (can't find namespace)

203   Unknown attribute prefix (can't find namespace)

204   Invalid XML markup

205   Invalid end state for parser

206   Document contains unbalanced element

207   Invalid XPath

208   No such child

209   Top element does not match start of path

210   DOM tree unavailable

302   Can't open file.

4.3.4.227 @JSONXPATH

@JSONXPATH[["*filename*",]*path*] : JSON XPath query.

If *filename* is not specified, @JSONXPATH will use the current JSON file opened by @JSONOPEN.

The *path* is a series of one or more element accessors separated by '/'. The path can be absolute (starting with '/') or relative to the current XPath location. Note: When using XPath notation the root element is always referred to as "json". This means all paths will begin with "/json".

The following are possible values for an element accessor:

name	A particular element name.
[i]	The i-th subelement of the current element.
..	the parent of the current element.

Example:

For example, with this JSON file:

```
{
  "firstlevel": {
    "one": "value",
    "two": ["first", "second"],
    "three": "value three"
  }
}

echo %@jsonxpath["test.json",/json/firstlevel/one/]
"value"

echo %@jsonxpath["test.json",/json/firstlevel/two/[2]/]
"second"
```

JSON Errors

10231　Unbalanced element tag

10232　Invalid JSON markup

10233　Invalid XPath

10234　DOM tree unavailable

XMLp Errors

101　Invalid attribute index

102　No attributes available

103　Invalid namespace index

104　No namespaces available

105　Invalid element index

106　No elements available

107　Attribute does not exist

201　Unbalanced element tag

202　Unknown element prefix (can't find namespace)

203　Unknown attribute prefix (can't find namespace)

204　Invalid XML markup

205　Invalid end state for parser

206　Document contains unbalanced element

207　Invalid XPath

208　No such child

209　Top element does not match start of path

210　DOM tree unavailable

302　Can't open file.

4.3.4.228 @JUNCTION

@JUNCTION[*dir***]** : Returns the directory referenced by the specified junction.

Example:

```
mklink /j test2 test
Junction created for test2 <<===>> test

echo %@junction[test2]
test
```

4.3.4.229 @LABEL

@LABEL[*d:***]**: Returns the volume label of the specified disk drive. The drive letter must be followed by a colon.

If the argument is a partial filename without a drive, @LABEL will expand the filename to get the drive.

Examples:

```
echo %@label[C:]
```

```
Windows10

echo %@label[%_disk:]
Development
```

See also: [VOL](#).

4.3.4.230 @LCS

@LCS[string1,string2**]** : Returns a pointer to the Longest Common Subsequence in **string1** and **string2**.

Example:

```
echo %@lcs[First string,second string]
rst
```

4.3.4.231 @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:

```
echo %@LEFT[2,jpsoft]
jp

echo %@LEFT[22,jpsoft]
jpsoft

echo %@LEFT[-2,jpsoft]
jpso

echo "%@LEFT[-22,jpsoft]"
""
```

4.3.4.232 @LEN

@LEN[string**]** : Returns the length of **string**.

Examples:

```
echo %@len[this is a test]
14

echo %@len[%comspec]
41
```

4.3.4.233 **@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 for additional details.

If **filename** is quoted, the returned filename will also be quoted (if necessary).

Example:

```
echo "%@lfn[c:\progra~1]"
"C:\Program Files"
```

4.3.4.234 **@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 (a sequence of 0-9 digits) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

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 command, or @FILEOPEN and a sequence of @FILEREADs.

You can retrieve input from standard input if you specify **CON** as the filename. If you are redirecting input to @LINE using this feature, you must use command grouping or the redirection will not work properly (you can pipe to @LINE without a command group; this restriction applies only to input redirection). For example:

```
(echo %@line[con,0]) < myfile.dat
```

@LINE can retrieve data from file streams on NTFS drives if the stream name is specified. See NTFS File Streams for additional details on file streams.

@LINE supports the TCC-RT clipboards (CLIP0: - CLIP9:) and temporary files (TMP0: - TMP9:).

Beware of characters with special meaning to TCC-RT, such as redirection and piping symbols, within the file. Use SETDOS /X with appropriate codes as needed.

4.3.4.235 **@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 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 for additional details on file streams.

@LINES supports the TCC-RT clipboards (CLIP0: - CLIP9:) and temporary files (TMP0: - TMP9:).

@LINES also sets two variables:

%_LINES_MAXLEN	The length of the longest line (in characters)
%_LINES_MAXLOC	The line number (base 0) of the longest line

Example:

```
echo %@lines[readme.txt]
170
```

4.3.4.236 @LINKS

@LINKS[*filename*] : Returns the number of hard links for the specified file (NTFS only).

@LINKS may not work for remote files (depending on your network redirector and the type of server you are querying).

See also MKLNK.

Example:

```
echo %@links[c:\windows\explorer.exe]
2
```

4.3.4.237 @LOWER

@LOWER[string**]** : Returns the **string** converted to lower case.

Examples:

```
echo %@lower[ThiS iSS aTeSt]
this is a test

echo %@lower[%path]
c:\windows\system32
```

4.3.4.238 @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"
```

4.3.4.239 @LUA

@LUA[*expression***]** : Execute a Lua expression.

Examples:

```
echo %@lua[print 'foo']
foo
```

4.3.4.240 @MACADDRESS

@MACADDRESS[*IPaddress***]**: Returns the unique Media Access Control (MAC) address of the network interface at *IPAddress*. An invalid or unknown address will return an error (see @ERRTEXT to decipher the error number if necessary).

See also @IPADDRESS.

Example:

```
echo %@macaddress[192.168.1.2]
00-7e-18-d5-2d-09
```

4.3.4.241 @MAKEAGE

@MAKEDATE[*n[,d]***]**: Returns a date, formatted according to the current country settings, or as explicitly specified by ***d*** (see Date Display Formats). ***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. 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 8601 (yyyy-mm-dd)
5	ISO 8601 (yyyy-Www-d)
6	ISO 8601 (yyyy-ddd)

Examples:

```
echo %@makedate[7924]
09/11/01

echo %@makedate[7924,4]
2001-09-11
```

4.3.4.242 @MAKEDATE

@MAKEDATE[_n[,d]_**]**: Returns a date, formatted according to the current country settings, or as explicitly specified by _d_ (see Date Display Formats). _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. 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 8601 (yyyy-mm-dd)
5	ISO 8601 (yyyy-Www-d)
6	ISO 8601 (yyyy-ddd)

Examples:

```
echo %@makedate[7924]
09/11/01

echo %@makedate[7924,4]
2001-09-11
```

4.3.4.243 @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.

Examples:

```
echo %@maketime[45240]
12:34:00

echo %@maketime[79244]
22:00:44
```

4.3.4.244 @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 (a sequence of 0-9 digits) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

Example_:_

```
echo %@max[1,5,2,0,-1]
5
```

4.3.4.245 @MD5

String mode:	**@MD5[***s[a|8],string[,start[,length]]***]**
File mode:	**@MD5[***[f,]filename[,start[,length]]***]**
Binary mode:	**@MD5[***[b,]handle[,start[,length]]***]**

Returns the 32 hexadecimal digit MD5 hash of the characters in *string*, the contents of the file *filename*, or the contents of the binary buffer.

If the first parameter is *s* for a Unicode UTF16 string, *sa* for an ASCII string, or *s8* for a UTF8 string, 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.

If the first parameter is *b* for a binary buffer, *handle* is the handle returned by @BALLOC.

The optional *start* parameter specifies the starting position in the binary buffer or file (in bytes, defaulting to 0).

The optional *length* parameter specifies the length of the buffer to hash (in bytes, defaulting to 0).

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, @SHA384, @SHA512, and @CRC32.

Example:

```
echo %@md5[s,this is a string]
93D64091ADF43E8FC0B74257AFD82FC3
```

4.3.4.246 @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 (a sequence of 0-9 digits) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

Example*:*

```
echo %@min[1,5,2,0,−1]
−1
```

4.3.4.247 @MONTH

@MONTH[*date[,format]***]** : Returns the month number for the specified date (1-12). See date formats 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)
5	ISO 8601 yyyy-Www-d
6	ISO 8601 yyyy-ddd

Examples:

```
echo %@month[2018-01-01]
1

echo %@month[%_date]
5
```

4.3.4.248 @MX

@MX[*address*] : Returns the email server for the specified user address.

Example:

```
echo %@mx[support@jpsoft.com]
direct-connect.jpsoft.com
```

4.3.4.249 @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 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]
xyz

echo "%@name[%_comspec]"
"tcc"
```

4.3.4.250 @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.
3. The remainder of *string* must be composed entirely of decimal digits (0 to 9), the thousands separator, 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

4.3.4.251 @ODBCCLOSE

@ODBCCLOSE[] : Close the current ODBC database session.

Example:

@ODBCCLOSE[]

4.3.4.252 @ODBCOPEN

@ODBCOPEN[*name*] : Open a SQL database using the ODBC driver.

Example:

4.3.4.253 @ODBCQUERY

@ODBCQUERY[*arrayvar,* "*query*"**]** : Send a query to a SQL database through the ODBC driver. Returns the string result of the query.

arrayvar - An array variable that receives the output of the SQL query. (You must create it with SETARRAY before calling @ODBCQUERY.)

"query" - The SQL query to execute.

You must have called @ODBCOPEN or ODBC /O "*name*" before calling @ODBCQUERY.

4.3.4.254 @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 for details.

Examples:

```
echo %@option[passiveftp]
Yes

echo %@option[stdcolors]
0
```

4.3.4.255 @OWNER

@OWNER[filename**]**: Returns the owner of the specified file (if any).

Examples:

```
echo %@owner[c:\windows\explorer.exe]
NT SERVICE\TrustedInstaller

echo %@owner[v.exe]
ASUS-PC\Rex
```

4.3.4.256 @PARSE

@PARSE[line,switches[,arg]**]** : Parse the command line for switches, returning an OR'd value for matching switches or optionally the argument(s) following the switch.

line - The (double quoted) command line to parse. If line is ".", TCC-RT will substitute the command line for the current batch file.

switches - One or more switch arguments (for example, /RST will match either an /R, and /S, or a /T on the command line.

arg - An optional integer value for the argument(s) following the switch to return. A 0 will return the switch, 1 the first argument following the switch. A * will return the remainder of the command line following the switch.

4.3.4.257 @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 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 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:

```
echo "%@path["c:\program files\xyz.abc"]
"c:\program files\"

echo "%@path[xyz.abc]"
""
```

4.3.4.258 @PERL

@PERL[*expression***]** : Executes the specified Perl expression. @PERL requires PerlScript, the WSH COM interface to Perl. PerlScript is available with Active State Perl (from www.activestate.com).

4.3.4.259 @PID

@PID[*filename[,+]***]** : Returns the PID for specified name (or 0 if no match). If you have multiple copies of the same executable running, **@PID** will return the first one it finds. The *filename* argument supports wildcards.

If you specify the optional second argument **+**, @PID will return all PID's that match the first argument.

```
@pid[firefox.*,+]
```

See also: @PPID, @PIDCOMMAND, @PIDUSER.

Example:

```
echo %@pid[tcc.exe]
22420

echo %@pid[firefox.exe,+]
11317 9466 12440
```

4.3.4.260 @PIDCOMMAND

@PIDCOMMAND[*pid***]** : Returns the startup command line for the specified process ID.

Example:

```
set pid=%@pid[tcc.exe]
echo %@pidcommand[%pid]
"C:\Program Files\JPSoft\TCMD17x64\TCC.EXE"
```

4.3.4.261 @PIDUSER

@PIDUSER[*pid***]** : Returns the user name that owns the specified process ID.

System processes return an empty string.

Example:

```
echo %@piduser[2190]
rconn
```

4.3.4.262 @PING

@PING[*host[,timeout[,packetsize[,ttl[,type]]]]***] :** 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 (default is 60).
> ***packetsize*** (optional) is the size of the data packet sent to the host in the ping request (default is 64). The minimum packet size is 12 bytes, and the maximum is 65520 bytes.
> ***ttl*** (optional) is the time to live (default is the TTL value of the underlying TCP/IP subsystem).
> ***type*** (optional) is the ICMP service type (default is 8).

@PING supports either IPv4 or IPv6 addresses. IPv6 requires an elevated session.

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]
```

4.3.4.263 PINGR

@PINGR[*host[,timeout[,packetsize[,ttl[,type]]]]***] :** Returns the address of the host responding to the PING (ICMP ECHO) request.

> ***host*** is the IP address or name.
> ***timeout*** is the maximum number of seconds to wait (default is 60).
> ***packetsize*** (optional) is the size of the data packet sent to the host in the ping request (default is 64). The minimum packet size is 12 bytes, and the maximum is 65520 bytes.
> ***ttl*** (optional) is the time to live (default is the TTL value of the underlying TCP/IP subsystem).
> ***type*** (optional) is the ICMP service type (default is 8).

The host address returned by @PINGR may or may not be the host specified in the first argument.

@PINGR supports either IPv4 or IPv6 addresses. IPv6 requires an elevated session.

A negative value indicates an error. If the request times out, @PINGR returns **-1**. An unreachable host returns **-2**. An invalid address returns **-3**.

Examples:

```
echo %@pingr[microsoft.com]
20.81.111.85
```

4.3.4.264 @PLUGIN

@PLUGIN[*module***] :** Returns the full pathname for the specified plugin name.

Example:

```
echo %@plugin[plugin]
D:\TakeCommand28\x64\Debug\PlugIns\plugin.dll
```

4.3.4.265 @PLUGINVER

@PLUGINVER[*plugin***] :** Returns the version number (major.minor.build) for the specified plugin.

Example:

```
echo %@pluginver[plugin]
1.0.1
```

4.3.4.266 @PPID

@PPID[*name***] :** Returns the PID for the parent process of the specified name (or 0 if no match). If you have multiple copies of the same executable running, **@PPID** will return the parent PID for the first one it finds.

If the *name* argument begins with a **=**, it is assumed to be a PID instead of a process name.

Example:

```
echo %@ppid[tcc.exe]
21960
```

4.3.4.267 @PRIME

@PRIME[*n***] :** Returns the first prime number >= *n* (a 64-bit integer).

Example:

```
echo %@prime[13798225]
13798247
```

4.3.4.268 @PRIORITY

@PRIORITY[*pid***] :** Returns the priority class for the specified process ID. The return values are (in hex):

8000	Above normal
4000	Below normal
100	Realtime
80	High
40	Idle
20	Normal

Example:

```
echo %@priority[33900]
20
```

4.3.4.269 @PROCESSIO

@PROCESSIO[*pid, option***]** : Returns the I/O information for a process.

pid - The Process ID

option - The requested info:

0 - The number of read operations performed
1 - The number of write operations performed
2 - The number of I/O operations performed, other than read and write operations
3 - The number of bytes read
4 - The number of bytes written
5 - The number of bytes transferred during operations other than read and write operations

Examples:

```
echo %_pid
33472

echo %@processio[33472,0]
187

echo %@processio[33472,1]
767

echo %@processio[33472,4]
188229
```

4.3.4.270 @PROCESSTIME

@PROCESSTIME[*pid,n***] :** Return the process time as a fileage. *n* is the time to return:

0 - Start time
1 - End time
2 - Kernel mode time
3 - User mode time

Example:

```
echo %@processtime[33900]
129811263230521496
```

4.3.4.271 @PSHELL

@PSHELL[*expression***]** : Executes the specified PowerShell expression.

4.3.4.272 @PUNYDECODE

@PUNYDECODE[s,*string***] :** Decode a Punycode string.
@PUNYDECODE[*inputfile***,***outputfile***] :** Decode a Punycode file.

4.3.4.273 @PUNYENCODE

@PUNYENCODE[s,*string***] :** Encode a Punycode string.
@PUNYENCODE[*inputfile***,***outputfile***] :** Encode a Punycode file.

4.3.4.274 @PYTHON

@PYTHON[_expression_**]** : Executes the specified Python expression.

The Python interpreter in **_TCC-RT_** is persistent, so if you want to reset it pass an empty string to **@PYTHON**.

Example:

```
echo %@python[print("Printing from Python")]
Printing from Python
0
```

4.3.4.275 @QPDECODE

@QPDECODE[s,_string_**]** : Decode a string using the Quote-Printable MIME format (using only special characters).
@QPDECODE[_inputfile_,_outputfile_**]** : Decode a file using the Quote-Printable MIME format (using only special characters).

4.3.4.276 @QPENCODE

@QPENCODE[s,_string_**]** : Encode a string using the Quote-Printable MIME format (using only special characters).
@QPENCODE[_inputfile_,_outputfile_**]** : Encode a file using the Quote-Printable MIME format (using only special characters).

4.3.4.277 @QUOTE

@QUOTE[_string_**]** : Returns a double quoted argument if it contains any whitespace characters.

Examples:

```
echo %@quote[Now is the time]
"Now is the time"

echo %@quote[Nowisthetime]
Nowisthetime
```

4.3.4.278 @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-RT starts and will therefore produce a different sequence of numbers each time you use it. The maximum range between **min** and **max** is a signed 64-bit integer.

Examples:

```
echo %@random[0,1]
0

echo %@random[-10,10]
7

echo %@random[-10,10]
9
```

```
echo %@random[-10,10]
-6
```

4.3.4.279 @READSCR

@READSCR[*row,col,length***]**: Returns the text displayed in the TCC-RT 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.

4.3.4.280 @READY

@READY[*d:***]**: Returns **1** if the specified drive is ready; otherwise returns **0**. The drive letter must be followed by a colon.

If the argument is a partial filename without a drive, @READY will expand the filename to get the drive.

@READY does not support UNC names.

Examples:

```
echo %@ready[E:]
0

echo %@ready[%_boot:]
1
```

4.3.4.281 @REGBREAD

@REGBREAD[*HKEY...\subkey\value,handle,length***]:** Read a value from the registry to a binary buffer.

> *handle* : A buffer previously created with @BALLOC.
> *length :* The length (in bytes) to read from the registry key.

If @REGBREAD succeeds, it returns "0", otherwise it returns the Windows error number.

If you are running a 64-bit version of Windows, you can access the 64-bit registry instead of the 32-bit registry by appending "_64" to the HKEY name.

If the key name begins with *machinename*, @REGBREAD opens the registry on a remote machine. The remote registry service must be running on the remote machine, and you must have access and permissions. The HKEY parameter can be one of the following keys:

> HKEY_CLASSES_ROOT (or HKCR)
> HKEY_CURRENT_CONFIG (or HKCC)
> HKEY_CURRENT_USER (or HKCU)
> HKEY_LOCAL_MACHINE (or HKLM)
> HKEY_PERFORMANCE_DATA (or HKPD)

HKEY_USERS (or HKU)

Note: Remember to use quotes around any entry containing spaces or commas!

See @REGBWRITE and @BALLOC.

4.3.4.282 @REGBWRITE

@REGBWRITE[*HKEY...\subkey\value,type,handle,length***]:** Write a value from a binary buffer to the registry.

> *type :* The type of key. @REGBWRITE supports keys of type REG_BINARY and REG_NONE.
> *handle* : A buffer previously created with @BALLOC.
> *length :* The length (in bytes) to write to the registry key.

If @REGBWRITE succeeds, it returns "0", otherwise it returns the Windows error number.

If you are running a 64-bit version of Windows, you can access the 64-bit registry instead of the 32-bit registry by appending "_64" to the HKEY name.

If the key name begins with *machinename*, @REGBWRITE opens the registry on a remote machine. The remote registry service must be running on the remote machine, and you must have access and permissions. The HKEY parameter can be one of the following keys:

> HKEY_CLASSES_ROOT (or HKCR)
> HKEY_CURRENT_CONFIG (or HKCC)
> HKEY_CURRENT_USER (or HKCU)
> HKEY_LOCAL_MACHINE (or HKLM)
> HKEY_PERFORMANCE_DATA (or HKPD)
> HKEY_USERS (or HKU)

Note: Remember to use quotes around any entry containing spaces or commas!

See @REGBREAD and @BALLOC.

4.3.4.283 @REGCOPYKEY

@REGCOPYKEY[*HKEY...\key, targetkey***]** : Recursively copy the specified key and all of its subkeys to the target key. Returns **1** if the key was copied, **0** otherwise. The key names must be enclosed in double quotes if they contain any separator characters (space, comma, or tab).

Both keys must be in the same root (HKCR, HKCU, HKLM, HKU, or HKCC).

If you are running a 32-bit version of TCC-RT in a 64-bit version of Windows, you can access the 64-bit registry instead of the 32-bit registry by appending "_64" to the HKEY name.

If the key name begins with *machinename*, @REGCOPYKEY opens the registry on a remote machine. The remote registry service must be running on the remote machine, and you must have access and permissions. The HKEY parameter can be one of the following keys:

> HKEY_CLASSES_ROOT (or HKCR)
> HKEY_CURRENT_CONFIG (or HKCC)
> HKEY_CURRENT_USER (or HKCU)
> HKEY_LOCAL_MACHINE (or HKLM)
> HKEY_PERFORMANCE_DATA (or HKPD)

HKEY_USERS (or HKU)

Note: If you are copying a very large tree, this function can take several minutes to finish. (This is a Windows issue, not TCC-RT.)

See @REGCREATE for information on the format of the key name.

4.3.4.284 @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.

If you are running a 64-bit version of Windows, you can access the 64-bit registry instead of the 32-bit registry by appending "_64" to the HKEY name. For example:

```
@regcreate["HKLM_64\Software\Company\Product\User"]
```

If the key name begins with _machinename_, @REGCREATE opens the registry on a remote machine. The remote registry service must be running on the remote machine, and you must have access and permissions. The HKEY parameter can be one of the following keys:

HKEY_CLASSES_ROOT (or HKCR)
HKEY_CURRENT_CONFIG (or HKCC)
HKEY_CURRENT_USER (or HKCU)
HKEY_LOCAL_MACHINE (or HKLM)
HKEY_PERFORMANCE_DATA (or HKPD)
HKEY_USERS (or HKU)

See also: @REGQUERY (read a value), @REGSET (write a value), and @REGSETENV (write and broadcast a value).

4.3.4.285 @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).

If the key name begins with _machinename_, @REGDELKEY opens the registry on a remote machine. The remote registry service must be running on the remote machine, and you must have access and permissions. The HKEY parameter can be one of the following keys:

> HKEY_CLASSES_ROOT (or HKCR)
> HKEY_CURRENT_CONFIG (or HKCC)
> HKEY_CURRENT_USER (or HKCU)
> HKEY_LOCAL_MACHINE (or HKLM)
> HKEY_PERFORMANCE_DATA (or HKPD)
> HKEY_USERS (or HKU)

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 for information on the format of the key name.

Example:

```
echo %@regcreate["HKEY_CURRENT_USER\Software\JP Software\Take Command
28\foo"]
echo %@regdelkey["HKEY_CURRENT_USER\Software\JP Software\Take Command
28\foo"]
1
```

4.3.4.286 @REGEX

@REGEX[_expression,string_**]** : 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 for supported expressions.

Examples:

```
echo %@regex[\d,1234]
1

echo %@regex[\d,abcd]
0

echo %@regex[[b-chm-pP]at|ot,Pat]
1
```

4.3.4.287 @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 for supported expressions. (This function is basically a wildcard-enabled @INDEX.)

Examples:

```
echo %@regexindex[def,abcdefgh]
3

echo %@regexindex[\d,abcd1234]
4
```

4.3.4.288 @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).

If the key name begins with *machinename*, @REGEXIST opens the registry on a remote machine. The remote registry service must be running on the remote machine, and you must have access and permissions. The HKEY parameter can be one of the following keys:

> HKEY_CLASSES_ROOT (or HKCR)
> HKEY_CURRENT_CONFIG (or HKCC)
> HKEY_CURRENT_USER (or HKCU)
> HKEY_LOCAL_MACHINE (or HKLM)
> HKEY_PERFORMANCE_DATA (or HKPD)
> HKEY_USERS (or HKU)

See @REGCREATE for information on the format of the key name.

Example:

```
echo %@regexist["HKEY_CURRENT_USER\Software\JP Software\Take Command 28"]
1
```

4.3.4.289 @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 for supported expressions.

Examples:

```
echo %@regexsub[2,(\w+)\s(\1)\W,"She said that that was not correct."]
that

echo %@regexsub[0,(\w+\.)+\w+,"users.mail.com"]
users.mail.com

echo %@regexsub[1,(\w+\.)+\w+,"users.mail.com"]
mail.
```

4.3.4.290 @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.

If the key name begins with *machinename*, @REGQUERY opens the registry on a remote machine. The remote registry service must be running on the remote machine, and you must have access and permissions. The HKEY parameter can be one of the following keys:

        HKEY_CLASSES_ROOT (or HKCR)
        HKEY_CURRENT_CONFIG (or HKCC)
        HKEY_CURRENT_USER (or HKCU)
        HKEY_LOCAL_MACHINE (or HKLM)
        HKEY_PERFORMANCE_DATA (or HKPD)
        HKEY_USERS (or HKU)

Note: Remember to use quotes around any entry containing spaces or commas!

See @REGCREATE (create a subkey) for information on the format of the key name. See also: @REGSET (write a value) and @REGSETENV (write and broadcast a value).

Example:

```
echo %@regquery["HKCU\Software\JP Software\Take Command 28\Version"]
28.0.1.0
```

4.3.4.291 @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, REG_MULTI_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.

If you're setting REG_MULTI_SZ values, separate each _data_ argument with a comma.

If the key name begins with *machinename*, @REGSET opens the registry on a remote machine. The remote registry service must be running on the remote machine, and you must have access and permissions. The HKEY parameter can be one of the following keys:

        HKEY_CLASSES_ROOT (or HKCR)
        HKEY_CURRENT_CONFIG (or HKCC)
        HKEY_CURRENT_USER (or HKCU)
        HKEY_LOCAL_MACHINE (or HKLM)
        HKEY_PERFORMANCE_DATA (or HKPD)
        HKEY_USERS (or HKU)

Note: Remember to use quotes around any entry containing spaces or commas!

See @REGCREATE for information on the format of the key name. See also: @REGQUERY (read a value) and @REGSETENV (write and broadcast a value).

Example:

```
echo %@regset["HKCU\Software\JP Software\Take Command
28\MyVersion",REG_SZ,9999]
echo %@regquery["HKCU\Software\JP Software\Take Command 28\MyVersion"]
9999
```

4.3.4.292 @REGSETENV

@REGSETENV[*HKEY...\subkey\value,type,data***]** : The same as @REGSET, but a broadcast message is sent to all applications when the change is made, so that any application monitoring such messages can respond to the change immediately if it is designed to do so. @REGSETENV returns **0** if the value was written or the Windows error number if an error occurred.

If the key name begins with *machinename*, @REGSETENV opens the registry on a remote machine. The remote registry service must be running on the remote machine, and you must have access and permissions. The HKEY parameter can be one of the following keys:

  HKEY_CLASSES_ROOT (or HKCR)
  HKEY_CURRENT_CONFIG (or HKCC)
  HKEY_CURRENT_USER (or HKCU)
  HKEY_LOCAL_MACHINE (or HKLM)
  HKEY_PERFORMANCE_DATA (or HKPD)
  HKEY_USERS (or HKU)

Note: Remember to use quotes around any entry containing spaces or commas!

See @REGCREATE for information on the format of the key name. See also: @REGQUERY (read a value) and @REGSET (write a value).

Example:

```
echo %@regsetenv["HKCU\Software\JP Software\Take Command
28\MyVersion",REG_SZ,9999]
echo %@regquery["HKCU\Software\JP Software\Take Command 28\MyVersion"]
9999
```

4.3.4.293 @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)

If the key name begins with *machinename*, @REGTYPE opens the registry on a remote machine. The remote registry service must be running on the remote machine, and you must have access and permissions. The HKEY parameter can be one of the following keys:

  HKEY_CLASSES_ROOT (or HKCR)
  HKEY_CURRENT_CONFIG (or HKCC)

HKEY_CURRENT_USER (or HKCU)
HKEY_LOCAL_MACHINE (or HKLM)
HKEY_PERFORMANCE_DATA (or HKPD)
HKEY_USERS (or HKU)

Example:

```
echo %@regset["HKCU\Software\JP Software\Take Command
28\MyVersion",REG_SZ,9999]
echo %@regtype["HKCU\Software\JP Software\Take Command 28\MyVersion"]
1
```

4.3.4.294 @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.

If the argument is a partial filename without a drive, @REMOTE will expand the filename to get the drive.

Examples:

```
echo %@remote[e:]
1

echo %@remote[%_disk]
0
```

4.3.4.295 @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.

If the argument is a partial filename without a drive, @REMOVABLE will expand the filename to get the drive.

Examples:

```
echo %@removable[e:]
1

echo %@removable[%_disk]
0
```

4.3.4.296 @REPEAT

@REPEAT[*char,count***]** : Returns the character **char** repeated **count** times.

Numeric input may be entered in either decimal format (a sequence of 0-9 digits) 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
```

4.3.4.297 @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"]
"ftp://server/etc"

echo %@replace[^,,,A better, command processor]
A better command processor
```

4.3.4.298 @REREPLACE

@REREPLACE[*source_re,target_re,source***]** - Regular expression back reference replacement.

source_re - Regular expression to apply to the source

target_re - Regular expression for back reference

source - Source string

Example:

To replace the input string "a1.txt" with "1a.txt":

```
@REREPLACE[(.)(.)\.txt,\2\1.txt,a1.txt]
```

4.3.4.299 @REVERSE

@REVERSE[*string***]** : Reverses the order of the characters in ***string***.

Example:

```
echo %@reverse[Now is the time for all good men]
nem doog lla rof emit eht si woN
```

4.3.4.300 @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-RT expands variables and functions before passing ***expr*** to REXX.

If you want to return the result of the REXX expression, prefix the expression with a **=** or **return**. Otherwise, REXX will pass the result back to TCC-RT for evaluation.

Examples:

```
echo %@rexx[= 3 * 4]
set myprog=d:\path\xyz.exe
echo %@rexx[address(%@name[%myprog]); return address()]
```

Note: This function requires that an ooREXX (Object REXX) or Regina REXX interpreter be installed and properly configured. See REXX Support for more information on the REXX language.

4.3.4.301 @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

4.3.4.302 @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.

Example:

```
echo "%@rtrim[98XP,Windows XP]"
"Windows "
```

4.3.4.303 @RUBY

@RUBY[_expression_**]** : Returns the string result of the Ruby expression. Note that the Ruby environment is persistent within a _TCC-RT_ session, 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 internal variable, and the value returned by the last @RUBY call with the RUBYVALUE internal variable.

4.3.4.304 @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](#) command.

4.3.4.305 @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](#) 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](#) 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*.EXE_).

@SEARCH supports regular expressions in **_program_**.

Program and each directory specification in **_path_** must be in quotes if they contain white space or special characters. @SEARCH will add double quotes to the result if it contains whitespace 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]
"C:\Windows\system32\notepad.exe"

echo %@search[msv*.dll,"d:\my dir\"]
"D:\my dir\test\msvc.dll"
```

4.3.4.306 @SELECT

@SELECT[_filename,top,left,bottom,right,title[,sort[,startline[,keymask]]]_**]**: 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 string matching with wildcards or regular expressions.

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 the specified width is < 150, the **_top, left, bottom, right_** parameters are assumed to be rows/columns relative to the TCC-RT window. If the width is >= 150, the parameters are assumed to be screen coordinates (in pixels).

If you set the optional 7th parameter _sort_ to **1**, the list will be sorted alphabetically. If you set _sort_ to -1, the list will be sorted in reverse alphabetic order.

The optional 8th parameter *startline* specifies the line @SELECT should highlight at startup. (The first line is 1.) If you specify *startline*, you must also specify the *sort* parameter.

If you specify the optional 9th argument *keymask,* the searching is disabled, and TCC-RT will check input keystrokes for a match against the key mask. If a match is found, @SELECT will return the current line and set the _SELECT_KEY environment variable to the input key value. The key mask is in the same format as INKEY /K.

The selected line number will be returned in the SELECT_LINE environment variable (the first line is 1).

Return value:

- the text of the line the scrollbar is on if you press **Enter**
- an empty string if you press **Esc**.

Example:

```
call %@select["d:\path\my menu.txt",50,100,175,400,Select an option]
```

4.3.4.307 @SELECTARRAY

@SELECTARRAY[array,top,left,bottom,right,title[,sort[,startline[,keymask]]]**]**: Pops up a selection window with the elements of the specified 1-dimensional array variable, 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 string matching with wildcards or regular expressions.

If the specified width is < 150, the **top, left, bottom, right** parameters are assumed to be rows/columns relative to the TCC-RT window. If the width is >= 150, the parameters are assumed to be screen coordinates (in pixels).

If you set the optional 7th parameter *sorted* to **1**, the list will be sorted alphabetically.

The optional 8th parameter *startline* specifies the line @SELECT should highlight at startup. (The first line is 1.) If you specify *startline*, you must also specify the *sort* parameter.

If you specify the optional 9th argument *keymask,* the searching is disabled, and TCC-RT will check input keystrokes for a match against the key mask. If a match is found, @SELECT will return the current line and set the _SELECT_KEY environment variable to the input key value. The key mask is in the same format as INKEY /K.

The selected line number will be returned in the SELECT_LINE environment variable (the first line is 1).

Return value:

- the text of the line the scrollbar is on if you press **Enter**
- an empty string if you press **Esc**.

See also @SELECT.

4.3.4.308 @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.

If the argument is a partial filename without a drive, @SERIAL will expand the filename to get the drive.

Examples:

```
echo %@serial[C:]
1B:EB6D

echo %@serial[%_disk:]
F82B:746
```

See also: @LABEL.

4.3.4.309 @SERIALHW

@SERIALHW[*d:*]: Returns the hardware serial number of the specified disk drive. The drive letter must be followed by a colon.

If the argument is a partial filename without a drive, @SERIALFW will expand the filename to get the drive.

Examples:

```
echo %@serial[D:]
WD-WCC6Y4FTRH82
```

See also: @SERIAL and @LABEL.

4.3.4.310 @SERIALPORTCLOSE

@SERIALPORTCLOSE[*n*]: Close the serial port. *n* is the handle returned by a previous call to @SERIALPORTOPEN. Returns 0 if the close was successful, or 0 if it failed.

Example:

```
set port=%@serialportopen[com1,9600]
set string=%@serialportread[%port]
echo %@serialportwrite[%port,Goodbye!"
echo %@serialportflush[%port]
echo %@serialportclose[%port]
```

4.3.4.311 @SERIALPORTFLUSH

@SERIALPORTFLUSH[*n*]: Flush the contents of the serial port buffer. *n* is the handle returned by a previous call to @SERIALPORTOPEN. Returns 1 if the flush succeeded, or 0 if it failed.

Example:

```
set port=%@serialportopen[com1,9600]
set string=%@serialportread[%port]
echo %@serialportwrite[%port,Goodbye!"
```

```
echo %@serialportflush[%port]
echo %@serialportclose[%port]
```

4.3.4.312 @SERIALPORTOPEN

@SERIALPORTOPEN[*COMn*[, *baud*[, *parity*[, *bits*[, stopbits[, *flow*]]]]]] - Open a serial port for read & write. The parameters are:

COMn - The COM port to open (COM1 - COM9)

baud - The baud rate (110 - 256000)

parity - The parity scheme to use. This can be one of the following values:
no
odd
even
mark
space

bits - The number of bits in the bytes to transmit & receive

stopbits - The number of stop bits to be used. This can be one of the following values:

1
1.5
2

flow - The type of flow control to use. This can be one of the following values:
no
CtsRts
CtsDtr
DsrRts
DsrDtr
XonXoff

@SERIALPORTOPEN returns a handle to the serial port, which must be passed to the other serial port functions.

See also: @SERIALPORTCLOSE, @SERIALPORTFLUSH, @SERIALPORTREAD, @SERIALPORTWRITE.

Example:

```
set port=%@serialportopen[com1,9600]
set string=%@serialportread[%port]
echo %@serialportwrite[%port,Goodbye!"
echo %@serialportflush[%port]
echo %@serialportclose[%port]
```

4.3.4.313 @SERIALPORTREAD

@SERIALPORTREAD[*n*]: Return the contents of the serial port buffer. *n* is the handle returned by a previous call to @SERIALPORTOPEN.

Example:

```
set port=%@serialportopen[com1,9600]
set string=%@serialportread[%port]
echo %@serialportwrite[%port,Goodbye!"
echo %@serialportflush[%port]
echo %@serialportclose[%port]
```

4.3.4.314 @SERIALPORTWRITE

@SERIALPORTWRITE[n,text**]**: Writes a string to the serial port. *n* is the handle returned by a previous call to @SERIALPORTOPEN. Returns 1 if the write succeeded, or 0 if it failed.

Example:

```
set port=%@serialportopen[com1,9600]
set string=%@serialportread[%port]
echo %@serialportwrite[%port,Goodbye!"
echo %@serialportflush[%port]
echo %@serialportclose[%port]
```

4.3.4.315 @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
0x400000	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

4.3.4.316 @SERVICE

@SERVICE[*service,info***]** : Returns information about the specified service.

service - The service name to query. If the service doesn't exist, @SERVICE will return -1.

info - The information you want:

1 The type of service. This will return one or more of the following values:

- 1 Device driver
- 2 File system driver
- 16 The service runs in its own process
- 32 The service shares a process with other services
- 256 The service can interact with the desktop

2 The current state of the service. This will return one of the following values:

- 1 The service is not running
- 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

3 Returns the check-point value the service increments to report its progress during a lengthy start, stop, pause, or continue operation. This value will be 0 if there is no pending operation.

4 The control codes the service accepts and processes in its handler function. This will return a combination of the following values (you can check the return value with the @EVAL OR test):

- 1 The service can be stopped
- 2 The service can be paused and continued
- 4 The service is notified when system shutdown occurs
- 8 The service can reread its startup parameters without being stopped and restarted
- 16 The service is a network component that can accept changes in its binding without being stopped and restarted

32   The service is notified when the computer's hardware profile has changed
64   The service is notified when the computer's power status has changed
128   The service is notified when the computer's session status has changed
256   The service can perform pre-shutdown tasks

5   Returns the estimated time required for a pending start, stop, pause, or continue operation (in milliseconds).

Examples:

```
echo %@service[audiosrv,1]
16

echo %@service[audiosrv,2]
4

echo %@service[audiosrv,3]
0

echo %@service[audiosrv,4]
193
```

4.3.4.317 @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.

Example:

```
echo %@sfn[%comspec]
C:\PROGRA~1\JPSoft\TCMD31~1\TCC.EXE
```

4.3.4.318 @SHA1

String mode: **@SHA1[***s[a|8]*,*string[,start[,length]*]]**]
File mode: **@SHA1[[***f,]filename[,start[,length]*]]**]
Binary mode: **@SHA1[[***b,]handle[,start[,length]*]]**]

Returns the SHA1 checksum of the characters in **string**, the contents of the file **filename**, or the contents of the binary buffer. If the first parameter is **s** for a Unicode UTF16 string, **sa** for an ASCII string, or **s8** for a UTF8 string, any leading or trailing whitespace characters in **string** are included. If the first argument is a **b**, the **filename** argument should be the handle returned by @BALLOC.

The first parameter determines whether the output is in upper or lower case:

s or **f** or **b**   lower case
S or **F** or **B**   upper case

The optional **start** parameter specifies the starting position in the binary buffer or file (in bytes, defaulting to 0).

The optional *length* parameter specifies the length of the buffer to hash (in bytes, defaulting to 0).

Example:

```
echo %@sha1[c:\windows\notepad.exe]
7EB0139D2175739B3CCB0D1110067820BE6ABD29
```

See also @SHA256, @SHA384, @SHA512, @MD5, and @CRC32.

4.3.4.319 @SHA256

String mode: **@SHA256[***s[a|8]*,*string*[,*start*[,*length*]]**]**
File mode: **@SHA256[***[f,]filename[,start[,length]]*]**
Binary mode: **@SHA256[***[b,]handle[,start[,length]]*]**

Returns the SHA-256 checksum of the characters in *string*, the contents of the file *filename*, or the contents of the binary buffer. If the first parameter is *s* for a Unicode UTF16 string, *sa* for an ASCII string, or *s8* for a UTF8 string, any leading or trailing whitespace characters in *string* are included. If the first argument is a *b*, the *filename* argument should be the handle returned by @BALLOC.

The first parameter determines whether the output is in upper or lower case:

s or **f** or **b**   lower case
S or **F** or **B**   upper case

The optional *start* parameter specifies the starting position in the binary buffer or file (in bytes, defaulting to 0).

The optional *length* parameter specifies the length of the buffer to hash (in bytes, defaulting to 0).

Example:

```
echo %@sha256[c:\windows\notepad.exe]
142E1D688EF0568370C37187FD9F2351D7DDEDA574F8BFA9B0FA4EF42DB85AA2
```

See also @SHA384, @SHA512, @MD5, and @CRC32.

4.3.4.320 @SHA384

String mode: **@SHA384[***s[a|8]*,*string*[,*start*[,*length*]]**]**
File mode: **@SHA384[***[f,]filename[,start[,length]]*]**
Binary mode: **@SHA384[***[b,]handle[,start[,length]]*]**

Returns the SHA-384 checksum of the characters in *string*, the contents of the file *filename*, or the contents of the binary buffer. If the first parameter is *s* for a Unicode UTF16 string, *sa* for an ASCII string, or *s8* for a UTF8 string, any leading or trailing whitespace characters in *string* are included. If the first argument is a *b*, the *filename* argument should be the handle returned by @BALLOC.

The first parameter determines whether the output is in upper or lower case:

s or **f** or **b**   lower case
S or **F** or **B**   upper case

The optional *start* parameter specifies the starting position in the binary buffer or file (in bytes, defaulting to 0).

The optional *length* parameter specifies the length of the buffer to hash (in bytes, defaulting to 0).

Example:

```
echo %@sha384[c:\windows\notepad.exe]
04BA669372BD3CBC40CAA9E44DE7C2760DBC27D68A79F7B0DC24048D6FF7A883CC2F0A6A
B80AE6F4CD3E45045273873E
```

See also @SHA256, @SHA512, @MD5, and @CRC32.

4.3.4.321 @SHA512

String mode: **@SHA512[***s[a|8],string[,start[,length***]]]**
File mode: **@SHA512[***[f,]filename[,start[,length***]]]**
Binary mode: **@SHA512[***[b,]handle[,start[,length***]]]**

Returns the SHA-512 checksum of the characters in *string*, the contents of the file *filename*, or the contents of the binary buffer. If the first parameter is *s* for a Unicode UTF16 string, *sa* for an ASCII string, or *s8* for a UTF8 string, any leading or trailing whitespace characters in *string* are included. If the first argument is a *b*, the *filename* argument should be the handle returned by @BALLOC.

The first parameter determines whether the output is in upper or lower case:

s or **f** or **b**   lower case
S or **F** or **B**   upper case

The optional *start* parameter specifies the starting position in the binary buffer or file (in bytes, defaulting to 0).

The optional *length* parameter specifies the length of the buffer to hash (in bytes, defaulting to 0).

Example:

```
echo %@sha512[c:\windows\notepad.exe]
2F37A2E503CFFBD7C05C7D8A125B55368CE11AAD5B62F17AAAC7AAF3391A6886FA6A0FD7
3223E9F30072419BF5762A8AF7958E805A52D788BA41F61EB084BFE8
```

See also @SHA256, @SHA384, @MD5, and @CRC32.

4.3.4.322 @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 - <user name>\My Pictures
40 - USERPROFILE
41 - X86 system directory on x64
42 - x86 c:\Program Files on x64
43 - c:\Program Files\Common
44 - x86 Program Files\Common on x64
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

Examples:

```
echo %@shfolder[42]
C:\Program Files (x86)

echo %@shfolder[22]
C:\ProgramData\Microsoft\Windows\Start Menu
```

4.3.4.323 @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.

Example:

```
echo %@similar[now is the time,then was the time]
75
```

4.3.4.324 @SMCLOSE

@SMCLOSE[*handle* **]** - Close a handle to shared memory.

>**_handle_** - The handle returned by @SMOPEN

4.3.4.325 @SMOPEN

@SMOPEN[*size, name* **]** - Open and return a handle to shared memory.

>**_size_** - The size of shared memory (in bytes)

>**_name_** - The name of the shared memory. The name can have a "Global\" or "Local\" prefix to create the object in the global or session namespace. If the name is "Global\", then only an elevated session can open the shared memory.

4.3.4.326 @SMPEEK

@SMPEEK[*handle,offset,size*] : Read a value from shared memory.

>*handle* - a handle from @SMOPEN

>*offset* - the byte offset in the buffer (decimal or hex)

>*size* - the size of the value to read (in bytes):

>>1 - character
>>2 - short
>>4 - int
>>8 - int64

4.3.4.327 @SMPOKE

@SMPOKE[*handle,offset,size,value*] : Write a value to shared memory

>*handle* - a handle from @SMOPEN

>*offset* - the byte offset in the buffer (decimal or hex)

>*size* - the size of the value (in bytes):

>>1 - character
>>2 - short
>>4 - int
>>8 - int64

>*value* - the value to poke

@SMPOKE returns 0 on success.

4.3.4.328 @SMREAD

@SMREAD[*n,offset,type,length* **]** - Read a string to shared memory

n - The shared memory handle returned by @SMOPEN

offset - The offset (in bytes) from the beginning of the shared memory buffer.

type - Either **a** to read the string as ASCII or **u** to read it as Unicode.

length - The length to read (in characters).

4.3.4.329 @SMWRITE

@SMWRITE[*n,offset,type,string***]** : Write a string to shared memory

n - The shared memory handle returned by @SMOPEN

offset - The offset (in bytes) from the beginning of the shared memory buffer.

type - Either **a** to write the string as ASCII or **u** to write it as Unicode.

string - The string to write.

4.3.4.330 @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).

If the window argument is **DESKTOP**, the optional second argument specifies either which monitor (1 - *n*) whose desktop you want to save.

If the window argument is a window title, the optional second argument specifies whether you want the client area (0, the default) or the entire window (1) to be saved.

If the window argument begins with a **=**, it is assumed to be a PID instead of a window title.

@SNAPSHOT returns **0** if successful.

4.3.4.331 @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*.

4.3.4.332 @SUBSTR

@SUBSTR[_string,start,length_**]** : An older version of @INSTR. 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 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 (a sequence of 0-9 digits) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

Examples:

```
echo %@substr[this is useful,8]
useful

echo %@substr[this is useful,8,-2]
is

echo %@substr["commas, they DO matter",9]
they DO matter"

echo %@substr[commas^, they DO matter,9]
they DO matter
```

See also: @INSTR.

4.3.4.333 @SUBST

@SUBST[_n, string1, string2_**]**: Substitutes **string1** starting at position **n** in **string2**.

Numeric input may be entered in either decimal format (a sequence of 0-9 digits) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

4.3.4.334 @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]
```

4.3.4.335 @SYMLINK

@SYMLINK[_link_**]** : Returns the target referenced by the specified symbolic link.

Example:

```
symlink test2 test
Symlink created for test2 <<===>> test

echo %@sylink[test2]
test
```

4.3.4.336 @SYSTEMTIME

@SYSTEMTIME[_n_**]** **:** Return the system time as a [fileage](). _n_ is the time to return:

0 - The time that the system has been idle
1 - The time that the system has spent executing in Kernel mode (all threads in all processes, on all processors)
2 - The time that the system has spent executing in User mode (all threads in all processes, on all processors)

See also: [Time Stamps]().

Examples:

```
echo %@systemtime[0]
39709212923676

echo %@systemtime[1]
39709212923676

echo %@systemtime[2]
39709212923676
```

4.3.4.337 @TALNUM

@TALNUM[_string_**]**: Returns the number of alphabetic (a-z, A-Z) and/or numeric (0 - 9) characters in **_string_**.

See also: [@TALPHA](), [@TASCII](), [@TCNTRL](), [@TDIGIT](), [@TLOWER](), [@TPRINT](), [@TPUNCT](), [@TSPACE](), [@TUPPER](), and [@TXDIGIT]().

Example:

```
echo %@talnum[123abc]
6

echo %@talnum[123 abc]
6

echo %@talnum[1-2-3]
3
```

4.3.4.338 @TALPHA

@TALPHA[_string_**]**: Returns the number of alphabetic (a-z, A-Z) characters in **_string_**.

See also: @TALNUM, @TASCII, @TCNTRL, @TDIGIT, @TLOWER, @TPRINT, @TPUNCT, @TSPACE, @TUPPER, and @TXDIGIT.

Example:

```
echo %@talpha[abc123]
3

echo %@talpha[A B C]
3
```

4.3.4.339 @TARCOUNT

@TARCOUNT[_tararchive_**]**: Returns the number of files in a .tar archive.

See also TAR and UNTAR.

Example:

```
tar tcc *.cpp *.h
echo %@tarcount[tcc.tar]
147
```

4.3.4.340 @TARCFILE

@TARCFILE[_tararchive,n_**]**: Returns the compressed name of file _n_ in a .tar archive.

See also TAR and UNTAR.

Example:

```
tar tcc *.cpp
echo %@tarcfile[tcc.tar,1]
stdafx.cpp
```

4.3.4.341 @TARDFILE

@TARDFILE[_tararchive,n_**]**: Returns the decompressed name of file _n_ in a .tar archive.

See also [TAR](#) and [UNTAR](#).

Example:

```
tar /R tcc test\
echo %@tardfile[tcc.tar,4]
test\ntinit.cpp
```

4.3.4.342 @TARFILEDATE

@TARFILEDATE[_tararchive,n_**]**: Returns the date and time of file _n_ in a .tar archive.

See also [TAR](#) and [UNTAR](#).

```
tar tcc *.cpp
echo %@tarfiledate[tcc.tar,0]
2021-06-15 13:16:12
```

4.3.4.343 @TARFILESIZE

@TARFILESIZE[_tararchive,n_**]**: Returns the size of file _n_ in a .tar archive.

See also [TAR](#) and [UNTAR](#).

```
tar tcc *.cpp
echo %@tarfilesize[tcc.tar,0]
46868
```

4.3.4.344 @TASCII

@TASCII[_string_**]**: Returns the number of 7-bit ASCII characters (0x00 - 0x7F) in **_string_**.

See also: [@TALNUM](#), [@TALPHA](#), [@TCNTRL](#), [@TDIGIT](#), [@TLOWER](#), [@TPRINT](#), [@TPUNCT](#), [@TSPACE](#), [@TUPPER](#), and [@TXDIGIT](#).

Examples:

```
echo %@tascii[abc]
3

echo %@tascii[abc 123]
7

echo %@tascii["abc"a]
5
```

4.3.4.345 @TCL

@TCL[_expression_**]** : Returns the string result of the [Tcl](#) expression. (You cannot run a Tk script in @TCL, because there is no Tk event loop. If you want to run a Tk script, you need to execute it from the command line as you would a Tcl script, or with the @TK function.)

The Tcl interpreter in _**TCC-RT**_ is persistent, so if you want to reset it pass an empty string to @TCL.

See also [@TK](#).

4.3.4.346 @TCNTRL

@TCNTRL[_string_**]**: Returns the number of ASCII control characters (0x00 - 0x1F or 0x7F) in _**string**_.

See also: [@TALNUM](#), [@TALPHA](#), [@TASCII](#), [@TDIGIT](#), [@TLOWER](#), [@TPRINT](#), [@TPUNCT](#), [@TSPACE](#), [@TUPPER](#), and [@TXDIGIT](#).

Examples:

```
echo %@tcntrl[abc]
0

set var=^r^n
echo %@tcntrl[%var]
2
```

4.3.4.347 @TDIGIT

@TDIGIT[_string_**]** : Returns the number of digits (0-9) in _**string**_.

See also: [@TALNUM](#), [@TALPHA](#), [@TASCII](#), [@TCNTRL](#), [@TLOWER](#), [@TPRINT](#), [@TPUNCT](#), [@TSPACE](#), [@TUPPER](#), and [@TXDIGIT](#).

Example:

```
echo %@tdigit[0]
1

echo %@tdigit[123.456]
6

echo %@tdigit[-123]
3
```

4.3.4.348 @TIME

@TIME[_hh:mm:ss[am|pm]_**]** : Returns the number of seconds since midnight for the specified time. We recommend that you use a 24-hour time format for compatibility with all locales. If "am" or "pm" are specified @TIME will use a 12-hour format. 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]
45296

echo %@time[%_time]
81579
```

4.3.4.349 @TIMER

@TIMER[*n[,precision]***]** : Returns the current split time for a stopwatch started with the TIMER command. The value of **n** specifies the timer to read and can be 1 to 10.

The default @TIMER resolution is milliseconds (.001 seconds).

@TIMER accepts an optional second argument to return the timer split as a floating-point numeric value suitable for arithmetic. The possible values are:

ns	split time in nanoseconds
us	split time in microseconds
ms	split time in milliseconds
s	split time in seconds (3 digit decimal precision)
m	split time in minutes (5 digit decimal precision)
h	split time in hours (6 digit decimal precision)

Examples:

```
timer /1 on
echo %@timer[1]
0:00:.02.025

echo %@timer[1,h]
0.01468
```

4.3.4.350 @TK

@TK[*expression***]** : Returns the string result of the Tk expression. For example, this will execute the Tk script *test.tcl*:

> echo %@tk[source test.tcl]

Because of the way the Tk interpreter works, it is not possible for TCC-RT to maintain a persistent interpreter after executing a Tk script. TCC-RT will close the current Tcl/tk interpreter and create a new one the next time @TCL is executed.

See also @TCL.

4.3.4.351 @TLOWER

@TLOWER[*string***]** : Returns the number of lower case letters in **string**.

See also: @TALNUM, @TALPHA, @TASCII, @TCNTRL, @TDIGIT, @TPRINT, @TPUNCT, @TSPACE, @TUPPER, and @TXDIGIT.

Example:

```
echo %@tlower[hello]
5

echo %@tlower[Hello]
4
```

4.3.4.352 @TMP

@TMP[*n[,id]***]** : Returns line ***n*** from the TMP device. The first line is numbered 0. The string ****EOC**** is returned for all line numbers beyond the end of the TMP buffer.

The second parameter (0-9) specifies the TMP device you want to use (TMP0: - TMP9:).

Numeric input may be entered in either decimal format (a sequence of 0-9 digits) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

Examples:

```
echo %@tmp[0,0]

if "%@tmp[2,1]" eq "**EOC**" echo No more data in TMP1:

rem Return the first line in TMP7:
echo %@tmp[0,7]
```

4.3.4.353 @TMPWN

@TMPWN[*id, string***]** : Writes the ***string*** to the specified TMP device (0 - 9). Returns **0** if the operation was successful.

Examples:

```
if "%@tmpwn[2, save this line]" eq "0" echo Saved to TMP2:
Saved to TMP2:
```

4.3.4.354 @TPRINT

@TPRINT[*string***]**: Returns the number of printable characters in ***string***.

See also: @TALNUM, @TALPHA, @TASCII, @TCNTRL, @TDIGIT, @TLOWER, @TPUNCT, @TSPACE, @TUPPER, and @TXDIGIT.

Examples:

```
echo %@tprint[abc]
3

set var=abc^ndef
echo %@tprint[%var]
```

```
6
```

4.3.4.355 @TPUNCT

@TPUNCT[_string_**]**: Returns the number of punctuation characters in **_string_**, i.e. printable characters which are not alphanumeric or space.

See also: @TALNUM, @TALPHA, @TASCII, @TCNTRL, @TDIGIT, @TLOWER, @TPRINT, @TSPACE, @TUPPER, and @TXDIGIT.

Examples:

```
echo %@tpunct[.]
1

echo %@tpunct[+]
1

echo %@tpunct[:-)]
3

echo %@tpunct[.,a]
2
```

4.3.4.356 @TRIM

@TRIM[_string_**]** : Returns the string with the leading and trailing white space (space and tab characters) removed.

Example:

```
echo %@trim[  this is a test string       ]
this is a test string
```

4.3.4.357 @TRIMALL

@TRIMALL[_string_**]** : Returns the string with the leading and trailing white space (space and tab characters), and any extra internal white space removed.

Example:

```
echo %@trimall[  this is a          test                  string
 ]
this is a test string
```

4.3.4.358 @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.

A leading ~\ or ~/ will be interpreted as the current user's home directory.

If *filename* is quoted, the returned filename will also be quoted (if necessary).

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.

4.3.4.359 @TRUNCATE

@TRUNCATE[*handle***]** : Truncate the file opened for write access by @FILEOPEN at the current position of the file pointer, where *handle* is the value returned by @FILEOPEN.

See also the related handle-based functions:

@FILECLOSE	Close a file handle
@FILEOPEN	Open a file handle
@FILEREAD	Read next line from a file handle
@FILESEEK	Move a file handle pointer
@FILESEEKL	Move a file handle pointer to a specified line
@FILEWRITE	Write next line to a file handle
@FILEWRITEB	Write data to a file handle

4.3.4.360 @TSPACE

@TSPACE[*string***]**: Returns the number of white space characters (0x09 - 0x0D or 0x20) in *string*.

See also: @TALNUM, @TALPHA, @TASCII, @TCNTRL, @TDIGIT, @TLOWER, @TPRINT, @TPUNCT, @TUPPER, and @TXDIGIT.

Example:

```
echo %@tspace[ ]
3

echo %@tspace[hello world]
1
```

4.3.4.361 @TUPPER

@TUPPER[*string*] : Returns the number of upper case letters in *string*.

See also: @TALNUM, @TALPHA, @TASCII, @TCNTRL, @TDIGIT, @TLOWER, @TPRINT, @TPUNCT, @TSPACE, and @TXDIGIT.

Example:

```
echo %@tupper[hello]
0
```

```
echo %@tupper[Hello]
1
```

4.3.4.362 @TXDIGIT

@TXDIGIT[_string_**]**: Returns the number of characters in **_string_** that are hexadecimal digits (0-9 and A-F or a-f).

See also: @TALNUM, @TALPHA, @TASCII, @TCNTRL, @TDIGIT, @TLOWER, @TPRINT, @TPUNCT, @TSPACE, and @TUPPER.

Example:

```
echo %@txdigit[123abc]
6

echo %@txdigit[123 ttt]
3
```

4.3.4.363 @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).

4.3.4.364 @UNICODE

@UNICODE[_string_**]** : Returns the space separated list of the Unicode values of the characters in **_string_**. You can use the Escape character before a special character (i.e., a quote or greater than (>) sign) in **_string_**.

See also: @ASCII.

Examples:

function	value
%@unicode[a]	97
%@unicode[A]	65
%@unicode[^`]	96
%@unicode[abc]	97 98 99

4.3.4.365 @UNIQUE

@UNIQUE[*path[,prefix]***]** : Creates a zero-length file with a unique name in the specified directory, and returns its 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 **path** is quoted, the returned filename will also be quoted (if necessary).

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 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, DELAY /M, or BEEP (with a frequency less than 20 Hz) to provide a short delay between invocations. You may also be able to work around the problem by performing some disk I/O activity between invocations, as this can force physical creation of the file on the disk before @UNIQUE is invoked again.

Examples:

```
echo %@unique[d:\takecommand28]
D:\takecommand28\UNIE810.tmp

echo %@unique[d:\takecommand28,tc]
D:\takecommand28\tc725F.tmp
```

4.3.4.366 @UNQCLOSE

@UNQCLOSE[*filename***]** : Close a UnQlite database.

filename   Database opened with @UNQOPEN

Returns 0 if successful, or the error text if it fails.

Example:

```
set db=test.db
set result=%@unqopen[rwc,%db]
rem do some DB processing here ...
set result=%@unqclose[%db]
```

4.3.4.367 @UNQDELETE

@UNQDELETE[[*u,***]***filename,key***]** : Delete a key/value pair from a UnQlite database.

u   Optional flag that the key is Unicode (UTF16)
filename   Database opened with @UNQOPEN
key   Key to delete

Returns 0 if successful, or the error text if not.

Example:

```
echo Testing database delete
echo %@unqopen[rwc,test.db]
echo %@unqdelete[test.db,"key1"]
echo %@unqclose[test.db]
```

4.3.4.368 @UNQKVB

@UNQKVB[[*u,*]*filename,"key",bhandle[,length]*] : Add a key / binary blob value pair to a UnQlite database.

u　　　　　Optional flag that the key and value are Unicode (UTF16)
filename　　Database opened with @UNQOPEN
key　　　　Key to add or replace
bhandle　　Binary handle returned by @BALLOC
length　　　Optional length (in bytes) to write (if -1 or nor specified, write the entire buffer)

Returns 0 if successful, or the error text if not.

Example:

This example opens a database named "test.db", then allocates a binary buffer, writes it to the database, and then reads it back.

```
echo %@unqopen[rwc,test.db]
set handle=%@balloc[4096]
echo %@unqkvb[test.db,"bbb",%handle,-1]
echo %@unqreadb[test.db,"bbb",%handle,-1]
echo %@unqclose[test.db]
```

See also: @UNQOPEN, @UNQCLOSE, @UNQKVBA.

4.3.4.369 @UNQKVBA

@UNQKVBA[[*u,*]*filename,"key",bhandle[,length]*] : Append a binary blob to the value of an existing UnQlite key/value pair.

u　　　　　Optional flag that the key and value are Unicode (UTF16)
filename　　Database opened with @UNQOPEN
key　　　　Key to update
bhandle　　Binary handle returned by @BALLOC
length　　　Optional length (in bytes) to write (if -1 or nor specified, write the entire buffer)

Returns 0 if successful, or the error text if not.

See also: @UNQOPEN, @UNQCLOSE, @UNQKVB.

Example:

```
echo %@unqopen[rwc,test.db]
set handle=%@balloc[4096]
```

```
rem write something to the binary buffer
echo %@unqkvba[test.db,"bbb",%handle,-1]
echo %@unqclose[test.db]
```

4.3.4.370 @UNQKVF

@UNQKVF[[*u,*]*filename,"key","value"*] : Add a key / file value pair to a UnQLite database.

u   Optional flag that the key and value are Unicode (UTF16)
filename   Database opened with @UNQOPEN
key   Key to add or replace
filename   Store the contents of the file *filename* in the value
length   Length (in bytes) to write (or -1 for the entire file)

The maximum file size is dependent on the amount of RAM and disk space available.

Returns 0 if successful, or the error text if not.

Example:

This example opens a database named "test.db", writes a key/file to the database, and then reads it back, saving it in a new file.

echo Testing file write and read
echo %@unqopen[rwc,test.db]
echo %@unqkvf[test.db,"aaa",ansi.cmd]
echo %@unqreadf[test.db,"aaa",newansi.cmd]
echo %@unqclose[test.db]

See also: @UNQOPEN, @UNQCLOSE, @UNQKVFA.

4.3.4.371 @UNQKVFA

@UNQKVFA[[*u,*]*filename,"key",filename[,length]*] : Append the contents of a file to the value of an existing UnQlite key/value pair.

u   Optional flag that the key and value are Unicode (UTF16)
filename   Database opened with @UNQOPEN
key   Key to update
filename   File to append to the existing value
length   Optional length (in bytes) to write (if -1 or nor specified, write the entire file)

The maximum file size is dependent on the amount of RAM and disk space available.

Returns 0 if successful, or the error text if not.

See also @UNQKVF.

Example:

This example opens a database named "test.db", and appends the file *testfile.txt* to the value of key *aaa*.

```
echo Testing file write and read
```

```
echo This is a test file > testfile.txt
echo %@unqopen[rwc,test.db]
echo %@unqkvfa[test.db,"aaa",testfile.txt]
echo %@unqclose[test.db]
```

4.3.4.372 @UNQKVS

@UNQKVS[[*u,*]*filename,"key","value"*] : Add a key value pair to a UnQlite database.

u	Optional flag that the key and value are Unicode (UTF16)
filename	Database opened with @UNQOPEN
key	Key to add or replace
value	Value to add

Returns 0 if successful, or the error text if not.

Example:

This example opens a database named "test.db", writes a key and value to the database, and then reads it back.

```
echo Testing file write and read
echo %@unqopen[rwc,test.db]
echo %@unqkvs[test.db,"key1","This is the value for our first key"]
echo %@unqreads[test.db,"key1"]
echo %@unqclose[test.db]
```

See also: @UNQOPEN, @UNQCLOSE, @UNQKVSA.

4.3.4.373 @UNQKVSA

@UNQKVSA[[*u,*]*filename,"key","value"*] : Append to the value of an existing UnQlite key/value pair.

u	Optional flag that the key and value are Unicode (UTF16)
filename	Database opened with @UNQOPEN
key	Key to update
value	Value to append to the existing value

Returns 0 if successful, or the error text if not.

Example:

This example opens a database named "test.db", appends a string to an existing value, and then reads it back.

```
echo Testing file write and read
echo %@unqopen[rwc,test.db]
echo %@unqkvs[test.db,"key1","Append this to the first value"]
echo %@unqreads[test.db,"key1"]
echo %@unqclose[test.db]
```

4.3.4.374 @UNQOPEN

@UNQOPEN[*mode,filename*] : Open an UnQLite database. Use the same database name for the other @UNQ... functions.

The possible values for *mode* are:

RWC   Open a database with read+write privileges. The database is created if it doesn't exist.

RW   Open the database with read+write privileges. If the database does not exist, an error is returned.

RO   Open the database in read-only mode. If the database does not exist, an error is returned.

MM   A read-only memory-mapped view of the database.

If *filename* is ":mem:", then a private in-memory database is created. The in-memory database will be discarded when the database is closed.

If the specified database is already opened, @UNQOPEN will not open a new instance. So you cannot have the same database open with different read/write modes.

@UNQOPEN returns 0 if the database was successfully opened (or is already open), or non-zero on an error.

Example:

Open the database "test.db" :

```
set db=test.db
set result=%@unqopen[rwc,%db]
```

4.3.4.375 @UNQREADB

@UNQREADB[[*u,*]*filename,"key",handle[,length]*]: Read a binary value from an existing key in an UnQlite database.

u   Optional flag that the key is Unicode (UTF16)
filename   Database opened by @UNQOPEN
key   Key to read
handle   A binary handle returned by @BALLOC
length   Number of bytes to read. If not specified, read the entire binary buffer.

Returns 0 if successful, or an error if not.

Example:

Open the database "test.db", read the key "btest", and save the binary value to a binary buffer :

```
set db=test.db
set key=btest
set result=%@unqopen[rwc,%db]
REM Size unknown — let unqreadb expand buffer
set bhandle=%@balloc[1]
```

```
set result=%@unqreadb[%db,%key,%bhandle]
```

4.3.4.376 @UNQREADF

@UNQREADF[[*u,*]*filename,"key",outputname[,length]*] : Read a value from an existing key in a UnQLite database and save it to a file.

u	Optional flag that the key is Unicode (UTF16)
filename	Database opened with @UNQOPEN
key	Key to read
outputname	Output file that will contain the value
length	Optional length (in bytes) to read. If not specified, @UNQREADF will read the entire file.

Returns 0 if successful, or an error if not.

Example:

Open the database "test.db", read the key "btest", and save the value to the file "d:\temp\btest.value" :

```
set db=test.db
set key=btest
set result=%@unqopen[rwc,%db]
set result=%@unqreadf[%db,%key,"d:\temp\btest.value"]
```

4.3.4.377 @UNQREADS

@UNQREADS[[*u,*]*filename,"key"*] : Read a value from an existing key in an UnQlite database.

u	Optional flag that the key is Unicode (UTF16)
filename	Database opened with @UNQOPEN
key	Key to read

Returns the value as a string, or the error text. If the key doesn't exist (or doesn't have a value) **@UNQREADS** will not display anything.

Example:

Open the database "test.db", read the key "btest", and save the value to "result":

```
set db=test.db
set key=btest
set result=%@unqopen[rwc,%db]
set result=%@unqreads[%db,%key]
```

4.3.4.378 @UNQUOTE

@UNQUOTE[*string***]** : Returns the argument with all double quotes removed.

See also: @UNQUOTES

Example:

```
echo %@unquote["This is a """heavily" quoted" string"]
This is a heavily quoted string
```

4.3.4.379 @UNQUOTES

@UNQUOTES[*string***]** : Returns the argument with leading and trailing double quotes removed.

See also: @UNQUOTE

Example:

```
echo %@unquotes["This is a """heavily" quoted" string"]
This is a ""heavily" quoted" string
```

4.3.4.380 @UPPER

@UPPER[*string***]** : Returns ***string*** converted to upper case.

Example:

```
echo %@upper[this is a string]
THIS IS A STRING
```

4.3.4.381 @URLDECODE

@URLDECODE[*string***]** : Decode an URL encoded string, replacing %xx with the original characters.

4.3.4.382 @URLENCODE

@URLENCODE[*string***]** : Encode a string for Internet transmission, replacing non-alphanumeric characters with their %xx hex representation.

4.3.4.383 @USB

@USB[*d:***]** : Returns 1 if the specified drive letter is a USB drive.

Examples:

```
echo %@usb[d:]
0

echo %@usb[h:]
1
```

4.3.4.384 @UTF8DECODE

@UTF8DECODE[s,*string***]** : Create a text string (using the current code page) from a UTF8 input string. Returns the text string.
@UTF8DECODE[*inputfile,outputfile***]** : Decode a UTF8 encoded file. Returns 0 if the output file was successfully written.

4.3.4.385 @UTF8ENCODE

@UTF8ENCODE[s,*string***]** : Create a text string (using the current code page) from a UTF8 input string. Returns the text string.
@UTF8ENCODE[*inputfile,outputfile***]** : Encode a file from the current code page to UTF8. Returns 0 if the output file was successfully written.

Example:

```
echo %@utf8encode[s,This is a UTF8 string]
This is a UTF8 string
```

4.3.4.386 @UUDECODE

@UUDECODE[*inputfile,outputfile***]** : Decode a UU encoded file. Returns 0 if the output file was successfully written.

See also: @UUENCODE

4.3.4.387 @UUENCODE

@UUENCODE[*inputfile,outputfile***]** : Encode a UU encoded file. (3 bytes are encoded into 4 readable characters.) Returns 0 if the output file was successfully written.

See also: @UUDECODE

4.3.4.388 @UUID

@UUID[*n***]** : Returns a UUID (same as a GUID in Windows). *n* can be:

    0 - returns the UUID with lower case alphabetic characters and embedded hyphens
    1 - returns the UUID with upper case alphabetic characters and embedded hyphens
    2 - returns the UUID with lower case alphabetic characters and no hyphens
    3 - returns the UUID with upper chase alphabetic characters and no hyphens

Examples:

```
echo %@uuid[0]
2a1f64b4-d2cd-4e1b-accf-60effe6065f2

echo %@uuid[1]
94D8C597-5DD9-4947-95B5-B80B9EA223A0

echo %@uuid[2]
d9ccee3db2ad408fadea484cf8bdc977

echo %@uuid[3]
8A0EC71BD8ED4D1B986D071DF96426AE
```

4.3.4.389 @VARTYPE

@VARTYPE[*var*] : Returns the type (if any) for the specified variable name. The possible values are:

    0   No type
    1   Integer (0-9)
    2   Decimal (0-9, the decimal character, and the thousands separator)
    3   Hex (0-9, A-F)
    4   Boolean (0 or 1)
    5   Alphabetic (A-Z and a-z)
    6   Alphanumeric (A-Z, a-z, and 0-9)
    7   Regular expression

Variable types are set with the SET /T:*type* option.

Example:

```
set /t:2 tempvar=42
echo %@vartype[tempvar]
2
```

4.3.4.390 @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
```

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.

Examples:

```
echo %@verinfo[tcmd.exe,companyname,040904E4]
JP Software

echo %@verinfo[tcmd.exe,fileversion]
28.0.1
```

4.3.4.391 @VERSION

@VERSION[*filename[,separator[,start[,force][,prefix]]]***]** : Returns a serially "versioned" replacement for the file name. If the file doesn't exist, and *force* isn't set, @VERSION returns *filename.*

If **filename** is quoted, the returned filename will also be quoted (if necessary).

This is distinct from the function of @UNIQUE[] in that it retains the entire filename and only appends a version separator character and an ascending version number to the filename. @VERSION does not create the file; it just returns the next available version name.

@VERSION has four arguments:

> **filename**   The filename to "versionize" (required)

separator	The version separation character (optional, defaults to ';'). Note that the TCC-RT include list character is **;**, so if you want to use **;** in a filename, you will need to double quote the filename.
start	The starting version number (if necessary to add a version number; optional, defaults to '1')
force	The flag to force versioning, even if the file doesn't exist (optional, defaults to 0 or FALSE).
prefix	If *prefix* **is 0,** @VERSION will append the version number to the end of the extension. If *prefix* **is 1,** @VERSION will prefix the version number to the extension. (Optional, defaults to 0).

Examples:

```
echo %@version[myfile.txt]
myfile.txt;1

echo %@version[myfile.txt]
myfile.txt;2
```

4.3.4.392 @WATTRIB

@WATTRIB[*filename[,-attributes[,p]]***]**: If you do not specify any attributes, @WATTRIB returns the attributes of the specified file in the format **RHSADECIJNOFTVPU**, rather than **0** or **1**. If two or more parameters are specified, @WATTRIB returns a **1** if the specified file has the matching attribute(s); otherwise it returns a **0**. If the optional third argument **,p** is included (partial match), then @WATTRIB will return **1** if any of the attributes match

This function is similar to @ATTRIB, but supports file selection based on the following extended attributes available on NTFS volumes.

- **E**   Encrypted
- **N**   Normal
- **T**   Temporary
- **F**   Sparse file
- **J**   Junction or symbolic links
- **L**   Junction or symbolic links
- **C**   Compressed
- **O**   Offline
- **I**   Not content-indexed
- **V**   Virtualized
- **P**   Pinned
- **U**   Unpinned

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, **%@WATTRIB[SECURE.DAT]** would return RH_A_____. If the file does not exist, @WATTRIB returns an empty string.

The attributes (other than **N**) can be combined (for example %@ATTRIB[MYFILE,HS]). For example, **%@WATTRIB[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 (and no extended attributes).

Filename must be in quotes if it contains white space or special characters.

See also: Attributes Switches and the ATTRIB command.

Examples:

```
echo %@wattrib[tcmd.exe]
___A_____

echo %@wattrib[tcmd.exe,r]
0

echo %@wattrib[tcmd.exe,a]
1
```

4.3.4.393 @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. 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 **\util** anywhere	1
%path	*c	string ending with **c**	0
%path	*t	string ending with **t**	1
%path	c*	string starting with **c**	1
%path	t*	string starting with **t**	0
%path	*c*	string containing **c**	1
%path	*t*	string containing **t**	1
%path	*b*	string containing **b**	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

4.3.4.394 @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.

4.3.4.395 @WINCLASS

@WINCLASS[classname**] :** Returns the window title of the first window with the specified class name, or an empty string if no windows match.

Example:

```
echo %@winclass[consolewindowclass]
TCC Prompt
```

4.3.4.396 @WINCLIENTSIZE

@WINCLIENTSIZE[title] : Returns the client window size in the format height,width

If the title argument begins with a **=**, it is assumed to be a PID instead of a window title.

Example:

```
echo %@winclientsize[tc 28*]
1027,1634
```

4.3.4.397 @WINEXENAME

@WINEXENAME[title**]**: Returns the executable name for the first window matching **title** (which can include wildcards), or an empty string if none.

If the title argument begins with a **=**, it is assumed to be a PID instead of a window title.

Example:

```
echo %@winexename[tc 28*]
C:\Program Files\JPSoft\TCMD28\tcmd.exe
```

4.3.4.398 @WININFO

@WININFO[*n***]**: Returns information about the current system. ***n*** is a number specifying what information to return:

n	Information returned
1	Processor architecture 0 INTEL 6 IA64 9 **x**64 (AMD or Intel)
2	Processor bit mask (set of configured processors)
3	Number of processors
4	Type of processor **586** Pentium: **2200** Intel IA64 **8664** AMD or Intel x64
5	Processor level
6	Processor revision
7	page size, bytes
8	virtual memory allocation granularity, bytes

Examples:

```
echo @WININFO[1]
9
```

```
echo @WININFO[2]
255
```

```
echo @WININFO[3]
8
```

```
echo @WININFO[4]
8664
```

```
echo @WININFO[5]
6
```

```
echo @WININFO[6]
24067
```

```
echo @WININFO[7]
4096
```

```
echo @WININFO[8]
65536
```

4.3.4.399 **@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:

n	*Information returned*
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

Examples:

```
echo %@winmemory[0]
51

echo %@winmemory[1]
34303373312

echo %@winmemory[2]
16620326912

echo %@winmemory[3]
75105562624

echo %@winmemory[4]
37333237760

echo %@winmemory[5]
140737488224256

echo %@winmemory[6]
138533536821248
```

4.3.4.400 **@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.

n	*Information returned*
0	Width of screen on primary display monitor

1	Height of screen on primary display monitor
2	Width of the vertical scroll bar
3	Height of the 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 on primary display monitor
17	Height of client area for full-screen window on primary display monitor
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 horizontal 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
61	The default width, in pixels, of a maximized top-level window on the primary display monitor.

67	The value that specifies how the system is started:
	    0 Normal boot
	    1 Fail-safe boot
	    2 Fail-safe with network boot

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	

4.3.4.401 @WINPATH

@WINPATH[*filename*] **:** Convert a WSL filename format to Windows format. For example:

```
echo %@winpath[//mnt/c/windows/system32/notepad.exe]
c:\windows\system32\notepad.exe
```

4.3.4.402 @WINPID

@WINPID[*title*] : Returns the process ID for the window with the specified title, or -1 if no match is found.

Example:

```
echo %@winpid[TCC Prompt]
438636
```

4.3.4.403 @WINPOS

@WINPOS[*title***]:** Returns the screen coordinates of the window with the specified title, in the format "*top,left,bottom,right*".

If the *title* argument begins with a **=**, it is assumed to be a PID instead of a window title.

Example:

```
echo %@winpos[TCC Prompt]
25,25,367,702
```

4.3.4.404 @WINSIZE

@WINSIZE[*title*] : Returns the window size in the format *height,width*

If the *title* argument begins with a **=**, it is assumed to be a PID instead of a window title.

Example:

```
echo %@winsize[TCC Prompt]
342,677
```

4.3.4.405 @WINSTATE

@WINSTATE[*title***]** : Returns the window state of the first window matching ***title*** (which can include wildcards). The return values are:

Value	Window state
0	Hidden
1	Normal
2	Minimized
3	Maximized

If the *title* argument begins with a **=**, it is assumed to be a PID instead of a window title.

4.3.4.406 @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 (a sequence of 0-9 digits) 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).
120	121	The number of milliseconds a thread can go without dispatching a message before the system considers it unresponsive.
122	123	The number of milliseconds the system waits before terminating an application that does not respond to a shutdown request.
124	125	The number of milliseconds the service control manager waits before terminating a service that does not respond to a shutdown request.
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**

4.3.4.407 @WINTITLE

@WINTITLE[*pid***]**: Returns the window title of the process with the specified process ID.

Example:

```
echo %@wintitle[15380]
TCC Prompt
```

4.3.4.408 @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.

Examples:

```
%@wmi[root\cimv2,"SELECT name FROM Win32_Processor"]

%@wmi[root\cimv2,"SELECT name, state FROM Win32_service",4]
```

4.3.4.409 @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".

If you use a **-** and don't specify an end, @WORD will return all words from the *nth* one to the end of the line. For example:

    %@word[2-,A B C D E F G] will return "C D E F G".

The default list of separators for @FIELD, @FIELDS, @WORD and @WORDS 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. Alphabetic characters in *sep_list* are case sensitive.

@FIELD and @FIELDS differ from @WORD and @WORDS in how multiple consecutive separators are counted. @WORD and @WORDS consider a sequence as a single separator, and ignore separators at either end of string. In contrast, @FIELD and @FIELDS count each occurrence of a separator individually, including those at either end of string.

Numeric input may be entered in either decimal format (a sequence of 0-9 digits) 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**.

See also: @WORDS, @FIELD, @FIELDS.

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

4.3.4.410 @WORDS

@WORDS[[*"sep_list",]string*]: Returns the number of words in ***string***.

The default list of separators for @FIELD, @FIELDS, @WORD and @WORDS 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. Alphabetic characters in ***sep_list*** are case sensitive.

@FIELD and @FIELDS differ from @WORD and @WORDS in how multiple consecutive separators are counted. @WORD and @WORDS consider a sequence as a single separator, and ignore separators at either end of ***string***. In contrast, @FIELD and @FIELDS 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, @FIELD, @FIELDS.

Examples:

```
echo %@words[How many words in this list?]
6

echo %@words[How.many.words.in.this.list?]
1

echo %@words[".",How.many.words.in.this.list?]
6
```

4.3.4.411 @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. (To query a remote computer, you must be an authenticated user on that computer.)

4.3.4.412 @WSLPATH

@WSLPATH[_filename_**]:** Convert from the Windows filename format to WSL format. For example:

```
echo %@wslpath[c:\windows\system32\notepad.exe]
//mnt/c/windows/system32/notepad.exe
```

@WSLPATH will do a "truename" to see through network assignments, junctions, symbolic links, SUBST's, and home directory references.

4.3.4.413 @XMLCLOSE

@XMLCLOSE[] : Close an XML file previously opened by @XMLOPEN.

@XMLCLOSE returns 0 on success, or an XML error if it fails.

Example:

Bookstore.xml :

```
<?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>
```

Bookstore.btm:

```
@ECHO OFF
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[]
```

Running bookstore.btm outputs:

```
Harry Potter  costs only   29.99
Learning XML  costs only   39.95
Day Watch  costs only   14.99
Winston Churchill: An Autobiography  costs only   49.99
```

XML Errors:

> 101 Invalid attribute index
> 102 No attributes available
> 103 Invalid namespace index
> 104 No namespaces available
> 105 Invalid element index
> 106 No elements available
> 107 Attribute does not exist
> 201 Unbalanced element tag
> 202 Unknown element prefix (can't find namespace)
> 203 Unknown attribute prefix (can't find namespace)
> 204 Invalid XML markup
> 205 Invalid end state for parser
> 206 Document contains unbalanced elements
> 207 Invalid XPath
> 208 No such child
> 209 Top element does not match start of path
> 210 DOM tree unavailable
> 302 Can't open file
> 401 Invalid XML would be generated
> 402 An invalid XML name has been specified

4.3.4.414 @XMLCREATE

@XMLCREATE[*filename*] : Create an XML file for use by other XML variable functions.

If an XML file is already open it will be closed before the new file is created. If the file already exists, @XMLCREATE will return an error.

Returns 0 on success, or an XML error code on failure.

Example:

To create this XML named *books.xml*:

```
<book xmlns='urn:loc.gov:books' xmlns:isbn='urn:ISBN:0-395-36341-6'>
<title>Cheaper by the Dozen</title>
<isbn:number>1568491379</isbn:number>
</book>
```

Use the code:

```
echo %@xmlcreate[books.xml]
echo %@xmlstartelement[book,urn:loc.gov:books]
echo %@xmlputattr[xmlns:isbn,"",urn:ISBN:0-395-36341-6]
echo %@xmlputelement[title,urn:loc.gov:books,Cheaper by the Dozen]
echo %@xmlputelement[isbn:number,urn:ISBN:0-395-36341-6,1568491379);
echo %@xmlendelement[]
echo %@xmlclose[]
```

XML Errors:

```
101 Invalid attribute index
102 No attributes available
103 Invalid namespace index
104 No namespaces available
105 Invalid element index
106 No elements available
107 Attribute does not exist
201 Unbalanced element tag
202 Unknown element prefix (can't find namespace)
203 Unknown attribute prefix (can't find namespace)
204 Invalid XML markup
205 Invalid end state for parser
206 Document contains unbalanced elements
207 Invalid XPath
208 No such child
209 Top element does not match start of path
210 DOM tree unavailable
302 Can't open file
401 Invalid XML would be generated
402 An invalid XML name has been specified
```

4.3.4.415 @XMLENDELEMENT

@XMLENDELEMENT[] : Writes the closing tab of an XML element opened using @XMLSTARTELEMENT.

If no elements are open, @XMLENDELEMENT returns an error; if successful it returns 0.

Example:

To create this XML named *books.xml*:

```
<book xmlns='urn:loc.gov:books' xmlns:isbn='urn:ISBN:0-395-36341-6'>
<title>Cheaper by the Dozen</title>
<isbn:number>1568491379</isbn:number>
</book>
```

Use the code:

```
echo %@xmlcreate[books.xml]
echo %@xmlstartelement[book,urn:loc.gov:books]
echo %@xmlputattr[xmlns:isbn,"",urn:ISBN:0-395-36341-6]
echo %@xmlputelement[title,urn:loc.gov:books,Cheaper by the Dozen]
echo %@xmlputelement[isbn:number,urn:ISBN:0-395-36341-6,1568491379);
echo %@xmlendelement[]
echo %@xmlclose[]
```

XML Errors:

101 Invalid attribute index
102 No attributes available
103 Invalid namespace index
104 No namespaces available
105 Invalid element index
106 No elements available
107 Attribute does not exist
201 Unbalanced element tag
202 Unknown element prefix (can't find namespace)
203 Unknown attribute prefix (can't find namespace)
204 Invalid XML markup
205 Invalid end state for parser
206 Document contains unbalanced elements
207 Invalid XPath
208 No such child
209 Top element does not match start of path
210 DOM tree unavailable
302 Can't open file
401 Invalid XML would be generated
402 An invalid XML name has been specified

4.3.4.416 @XMLFLUSH

@XMLFLUSH[] : Flushes the XML parser buffers, and checks its end state.

@XMLFLUSH returns 0 on success, or an XML error if it fails.

XML Errors:

101 Invalid attribute index
102 No attributes available
103 Invalid namespace index
104 No namespaces available
105 Invalid element index
106 No elements available
107 Attribute does not exist
201 Unbalanced element tag
202 Unknown element prefix (can't find namespace)
203 Unknown attribute prefix (can't find namespace)
204 Invalid XML markup
205 Invalid end state for parser
206 Document contains unbalanced elements
207 Invalid XPath
208 No such child
209 Top element does not match start of path
210 DOM tree unavailable
302 Can't open file
401 Invalid XML would be generated
402 An invalid XML name has been specified

4.3.4.417 @XMLGETATTR

@XMLGETATTR[*["filename",],attributename*] : Returns the value of the specified attribute.

If you do not specify a filename, @XMLGETATTR will use the file previously opened by @XMLOPEN.

You must set the XPath before calling @XMLGETATTR.

Example:

Bookstore.xml :

```
<?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>
```

Bookstore.btm:

```
    @ECHO OFF
    SET a=%@XMLOPEN[bookstore.xml]
    SET b=%@XMLNODES[/bookstore]
    DO i = 1 to %b
            SET Title=%@XMLXPATH[/bookstore/book[%i]/title]
            SET Language=%@XMLGETATTR[lang]
            SET Price=%@XMLXPATH[/bookstore/book[%i]/price]
            ECHO %Title (in %Language) costs only %Price
    ENDDO
    SET c=%@XMLCLOSE[]
```

Running bookstore.btm outputs:

```
    Harry Potter (in jap) costs only 29.99
    Learning XML (in eng) costs only 39.95
    Day Watch (in ger) costs only 14.99
    Winston Churchill: An Autobiography (in eng) costs only 49.99
```

XML Errors:

101 Invalid attribute index
102 No attributes available
103 Invalid namespace index
104 No namespaces available
105 Invalid element index
106 No elements available
107 Attribute does not exist
201 Unbalanced element tag
202 Unknown element prefix (can't find namespace)
203 Unknown attribute prefix (can't find namespace)
204 Invalid XML markup
205 Invalid end state for parser
206 Document contains unbalanced elements
207 Invalid XPath
208 No such child
209 Top element does not match start of path
210 DOM tree unavailable
302 Can't open file
401 Invalid XML would be generated
402 An invalid XML name has been specified

4.3.4.418 @XMLHASXPATH

@XMLHASXPATH[["filename",],xpath] : Returns 1 if the *xpath* exists in the XML file, or 0 if it doesn't.

If you do not specify a filename, @XMLHASXPATH will use the file previously opened by @XMLOPEN.

@XMLHASXPATH may be used to check if an *xpath* exists before setting it via @XMLXPATH.

The XML parser in TCC-RT implements a subset of the XML XPath specification, allowing you to point to specific elements in the XML documents. The *xpath* is a series of one or more element accessors separated by '/'. The path can be absolute (starting with '/') or relative to the current XPath location.

The following are possible values for an element accessor:

'name'	A particular element name
name[i]	The i-th subelement of the current element with the given name
[i]	The i-th subelement of the current element
[last()]	The last subelement of the current element
[last()-i]	The subelement located at the last location minus i in the current element
name[@attrname="attrvalue"]	The subelement containing a particular value for a given attribute (supports single AND double quotes)
..	The parent of the current element

Example:

Bookstore.xml :

```
<?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>
```

Return 1 if the XPath exists:

```
ECHO %@XMLHASXPATH["bookstore.xml",/bookstore]
1
```

XML Errors:

101 Invalid attribute index
102 No attributes available
103 Invalid namespace index
104 No namespaces available
105 Invalid element index
106 No elements available
107 Attribute does not exist
201 Unbalanced element tag
202 Unknown element prefix (can't find namespace)
203 Unknown attribute prefix (can't find namespace)
204 Invalid XML markup
205 Invalid end state for parser
206 Document contains unbalanced elements
207 Invalid XPath
208 No such child
209 Top element does not match start of path
210 DOM tree unavailable
302 Can't open file
401 Invalid XML would be generated
402 An invalid XML name has been specified

4.3.4.419 @XMLNODENAMES

@XMLNODENAMES["*filename",]xpath*] : Returns a space delimited list of the element names for the specified xpath.

    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), **@XMLNODENAMES** will use the XML file previously opened by @XMLOPEN.

XML Errors:

    101 Invalid attribute index
    102 No attributes available
    103 Invalid namespace index
    104 No namespaces available
    105 Invalid element index
    106 No elements available
    107 Attribute does not exist
    201 Unbalanced element tag
    202 Unknown element prefix (can't find namespace)
    203 Unknown attribute prefix (can't find namespace)
    204 Invalid XML markup
    205 Invalid end state for parser
    206 Document contains unbalanced elements
    207 Invalid XPath
    208 No such child
    209 Top element does not match start of path
    210 DOM tree unavailable
    302 Can't open file
    401 Invalid XML would be generated

402 An invalid XML name has been specified

4.3.4.420 @XMLINPUT

@XMLINPUT[*inputdata*] : Parse an input string as XML data. (Use this instead of @XMLOPEN if you don't have an input file.)

Returns 0 on success, or an XML error code on failure.

***Examples*:**

```
echo %@xmlinput[<test>]
0

echo %@xmlinput[><><]
204
```

XML Errors:

101 Invalid attribute index
102 No attributes available
103 Invalid namespace index
104 No namespaces available
105 Invalid element index
106 No elements available
107 Attribute does not exist
201 Unbalanced element tag
202 Unknown element prefix (can't find namespace)
203 Unknown attribute prefix (can't find namespace)
204 Invalid XML markup
205 Invalid end state for parser
206 Document contains unbalanced elements
207 Invalid XPath
208 No such child
209 Top element does not match start of path
210 DOM tree unavailable
302 Can't open file
401 Invalid XML would be generated
402 An invalid XML name has been specified

4.3.4.421 @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.

Example:

Bookstore.xml :

```
<?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>
```

Bookstore.btm:

```
@ECHO OFF
SET a=%@XMLOPEN[bookstore.xml]
SET b=%@XMLNODES[/bookstore]
ECHO Total Nodes = %b
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[]
```

Running bookstore.btm outputs:

```
Total Nodes = 4
Harry Potter  costs only   29.99
Learning XML  costs only   39.95
Day Watch  costs only   14.99
Winston Churchill: An Autobiography  costs only   49.99
```

XML Errors:

    101 Invalid attribute index
    102 No attributes available
    103 Invalid namespace index
    104 No namespaces available
    105 Invalid element index

106 No elements available
107 Attribute does not exist
201 Unbalanced element tag
202 Unknown element prefix (can't find namespace)
203 Unknown attribute prefix (can't find namespace)
204 Invalid XML markup
205 Invalid end state for parser
206 Document contains unbalanced elements
207 Invalid XPath
208 No such child
209 Top element does not match start of path
210 DOM tree unavailable
302 Can't open file
401 Invalid XML would be generated
402 An invalid XML name has been specified

4.3.4.422 @XMLOPEN

@XMLOPEN[_filename_**]** - open an XML file for use by @XMLXPATH and/or @XMLNODES.

Returns 0 on success, or an XML error on failure;

Example:

Bookstore.xml :

```
<?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>
```

Bookstore.btm:

```
@ECHO OFF
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[]
```

Running bookstore.btm outputs:

```
Harry Potter  costs only   29.99
Learning XML  costs only   39.95
Day Watch  costs only   14.99
Winston Churchill: An Autobiography  costs only   49.99
```

XML Errors:

    101 Invalid attribute index
    102 No attributes available
    103 Invalid namespace index
    104 No namespaces available
    105 Invalid element index
    106 No elements available
    107 Attribute does not exist
    201 Unbalanced element tag
    202 Unknown element prefix (can't find namespace)
    203 Unknown attribute prefix (can't find namespace)
    204 Invalid XML markup
    205 Invalid end state for parser
    206 Document contains unbalanced elements
    207 Invalid XPath
    208 No such child
    209 Top element does not match start of path
    210 DOM tree unavailable
    302 Can't open file
    401 Invalid XML would be generated
    402 An invalid XML name has been specified

4.3.4.423 @XMLOUTPUT

@XMLOUTPUT[*data*] : Output XML to a string after processing.

Use this instead of @XMLSAVE if you created input with @XMLINPUT and you don't want to create a file.

Returns 0 on success, or an XML error on failure;

XML Errors:

    101 Invalid attribute index
    102 No attributes available
    103 Invalid namespace index
    104 No namespaces available
    105 Invalid element index
    106 No elements available

107 Attribute does not exist
201 Unbalanced element tag
202 Unknown element prefix (can't find namespace)
203 Unknown attribute prefix (can't find namespace)
204 Invalid XML markup
205 Invalid end state for parser
206 Document contains unbalanced elements
207 Invalid XPath
208 No such child
209 Top element does not match start of path
210 DOM tree unavailable
302 Can't open file
401 Invalid XML would be generated
402 An invalid XML name has been specified

4.3.4.424 @XMLPUTATTR

@XMLPUTATTR[*name,namespaceURI,value*] : Writes an XML attribute

@XMLPUTATTR writes an XML attribute on the currently opened XML element. It must be called right after calling @XMLSTARTELEMENT and before any calls to @XMLPUTSTRING, @XMLPUTCOMMENT, or @XMLPUTRAW. The file must have been opened with a previous @XMLOPEN.

If *name* is a local name without a prefix, the class will automatically introduce a new xmlns="NamespaceURI" attribute if necessary.

If *name* is in the form prefix:local, then class will automatically introduce a new xmlns:prefix="NamespaceURI" as necessary.

Certain attribute names will be handled in special ways by this method. If *name* is "xmlns" or uses the "xmlns" prefix, the attribute will be interpreted as a namespace declaration, regardless of the value of NamespaceURI. Similarly, any attribute using the "xml" prefix will be interpreted as a special attribute (like "xml:lang") and NamespaceURI will be ignored.

Returns 0 on success, or an XML error on failure.

Example:

To create this XML named *books.xml*:

```
<book xmlns='urn:loc.gov:books' xmlns:isbn='urn:ISBN:0-395-36341-6'>
<title>Cheaper by the Dozen</title>
<isbn:number>1568491379</isbn:number>
</book>
```

Use the code:

```
echo %@xmlcreate[books.xml]
echo %@xmlstartelement[book,urn:loc.gov:books]
echo %@xmlputattr[xmlns:isbn,"",urn:ISBN:0-395-36341-6]
echo %@xmlputelement[title,urn:loc.gov:books,Cheaper by the Dozen]
```

```
echo %@xmlputelement[isbn:number,urn:ISBN:0-395-36341-6,1568491379);
echo %@xmlendelement[]
echo %@xmlclose[]
```

XML Errors:

    101 Invalid attribute index
    102 No attributes available
    103 Invalid namespace index
    104 No namespaces available
    105 Invalid element index
    106 No elements available
    107 Attribute does not exist
    201 Unbalanced element tag
    202 Unknown element prefix (can't find namespace)
    203 Unknown attribute prefix (can't find namespace)
    204 Invalid XML markup
    205 Invalid end state for parser
    206 Document contains unbalanced elements
    207 Invalid XPath
    208 No such child
    209 Top element does not match start of path
    210 DOM tree unavailable
    302 Can't open file
    401 Invalid XML would be generated
    402 An invalid XML name has been specified

4.3.4.425 @XMLPUTCDATA

@XMLPUTCDATA[*text*] : Writes an XML CDATA block.

The file must have been opened with a previous @XMLOPEN.

Returns 0 on success, or an XML error on failure;

XML Errors:

    101 Invalid attribute index
    102 No attributes available
    103 Invalid namespace index
    104 No namespaces available
    105 Invalid element index
    106 No elements available
    107 Attribute does not exist
    201 Unbalanced element tag
    202 Unknown element prefix (can't find namespace)
    203 Unknown attribute prefix (can't find namespace)
    204 Invalid XML markup
    205 Invalid end state for parser
    206 Document contains unbalanced elements
    207 Invalid XPath
    208 No such child
    209 Top element does not match start of path
    210 DOM tree unavailable

302 Can't open file
401 Invalid XML would be generated
402 An invalid XML name has been specified

4.3.4.426 @XMLPUTCOMMENT

@XMLPUTCOMMENT[*text*] : Writes an XML comment block.

The file must have been opened with a previous @XMLOPEN.

Returns 0 on success, or an XML error on failure;

XML Errors:

101 Invalid attribute index
102 No attributes available
103 Invalid namespace index
104 No namespaces available
105 Invalid element index
106 No elements available
107 Attribute does not exist
201 Unbalanced element tag
202 Unknown element prefix (can't find namespace)
203 Unknown attribute prefix (can't find namespace)
204 Invalid XML markup
205 Invalid end state for parser
206 Document contains unbalanced elements
207 Invalid XPath
208 No such child
209 Top element does not match start of path
210 DOM tree unavailable
302 Can't open file
401 Invalid XML would be generated
402 An invalid XML name has been specified

4.3.4.427 @XMLPUTELEMENT

@XMLPUTELEMENT[*name,namespaceURI, value*] : Writes a simple XML element with no attributes and the specified value between the opening and closing tags.

If *name* is a local name without a prefix, TCC-RT will automatically introduce a new xmlns="NamespaceURI" attribute if necessary.

If *name* is in the form prefix:local, then TCC-RT will automatically introduce a new xmlns:prefix="NamespaceURI" as necessary.

When calling @XMLPutElement or @XMLStartElement, if a *namespaceURI* is not specified an empty namespace will be defined for the element. If a namespace should be associated with the element, a *namespaceURI* value must be provided. When creating the XML, TCC-RT will determine if the namespace already exists to avoid duplicate definitions of the same namespace.

Returns 0 on success, or an XML error on failure;

Example:

To create this XML:

```
<book xmlns='urn:loc.gov:books' xmlns:isbn='urn:ISBN:0-395-36341-6'>
<title>Cheaper by the Dozen</title>
<isbn:number>1568491379</isbn:number>
</book>
```

Use the code:

```
echo %@xmlstartelement[book,urn:loc.gov:books]
echo %@xmlputattr[xmlns:isbn,"",urn:ISBN:0-395-36341-6]
echo %@xmlputelement[title,urn:loc.gov:books,Cheaper by the Dozen]
echo %@xmlputelement[isbn:number,urn:ISBN:0-395-36341-6,1568491379);
echo %@xmlendelement[]
echo %@xmlclose[]
```

XML Errors:

    101 Invalid attribute index
    102 No attributes available
    103 Invalid namespace index
    104 No namespaces available
    105 Invalid element index
    106 No elements available
    107 Attribute does not exist
    201 Unbalanced element tag
    202 Unknown element prefix (can't find namespace)
    203 Unknown attribute prefix (can't find namespace)
    204 Invalid XML markup
    205 Invalid end state for parser
    206 Document contains unbalanced elements
    207 Invalid XPath
    208 No such child
    209 Top element does not match start of path
    210 DOM tree unavailable
    302 Can't open file
    401 Invalid XML would be generated
    402 An invalid XML name has been specified

4.3.4.428 @XMLPUTSTRING

@XMLPUTSTRING[*text*] : Writes text inside an XML element.

The XML file must have been opened with a previous @XMLOPEN.

Returns 0 on success, or an XML error on failure;

XML Errors:

    101 Invalid attribute index
    102 No attributes available
    103 Invalid namespace index

104 No namespaces available
105 Invalid element index
106 No elements available
107 Attribute does not exist
201 Unbalanced element tag
202 Unknown element prefix (can't find namespace)
203 Unknown attribute prefix (can't find namespace)
204 Invalid XML markup
205 Invalid end state for parser
206 Document contains unbalanced elements
207 Invalid XPath
208 No such child
209 Top element does not match start of path
210 DOM tree unavailable
302 Can't open file
401 Invalid XML would be generated
402 An invalid XML name has been specified

4.3.4.429 @XMLREMOVECHILDREN

@XMLREMOVECHILDREN[[*path*]] : Removes the children of the element at the specified (or current) XPath. The element itself remains.

If *xpath* is not specified, the current XPath is used.

Returns 0 on success, or an XML error on failure;

Example:

Starting with this XML:

```
<food>
    <fruits>
        <apple>
            <color:red</color.
        </apple>
        <banana>
            <color>yellow</color>
        </banana>
    </fruits>
</food>
```

To remove the children of the *apple* element while leaving *apple* in place:

```
echo %@xmlremovechildren[/food/fruits/apple]
0
```

The XML now looks like this:

```
<food>
    <fruits>
        <apple>
        </apple>
```

```
            <banana>
                    <color>yellow</color>
            </banana>
        </fruits>
</food>
```

XML Errors:

101 Invalid attribute index
102 No attributes available
103 Invalid namespace index
104 No namespaces available
105 Invalid element index
106 No elements available
107 Attribute does not exist
201 Unbalanced element tag
202 Unknown element prefix (can't find namespace)
203 Unknown attribute prefix (can't find namespace)
204 Invalid XML markup
205 Invalid end state for parser
206 Document contains unbalanced elements
207 Invalid XPath
208 No such child
209 Top element does not match start of path
210 DOM tree unavailable
302 Can't open file
401 Invalid XML would be generated
402 An invalid XML name has been specified

4.3.4.430 @XMLREMOVEELEMENT

@XMLREMOVEELEMENT[[*xpath*]] **:** Removes the element and its children at *xpath*.

If *xpath* is not specified, the current XPath is used.

Returns 0 on success, or an XML error on failure;

Example:

Starting with this XML:

```
<food>
    <fruits>
            <apple>
                    <color:red</color>
            </apple>
            <banana>
                    <color>yellow</color>
            </banana>
        </fruits>
</food>
```

To remove the *apple* element and all of its children:

```
echo %@xmlremoveelement[/food/fruits/apple]
0
```

The XML now looks like this:

```
<food>
    <fruits>
            <banana>
                    <color>yellow</color>
            </banana>
    </fruits>
</food>
```

XML Errors:

    101 Invalid attribute index
    102 No attributes available
    103 Invalid namespace index
    104 No namespaces available
    105 Invalid element index
    106 No elements available
    107 Attribute does not exist
    201 Unbalanced element tag
    202 Unknown element prefix (can't find namespace)
    203 Unknown attribute prefix (can't find namespace)
    204 Invalid XML markup
    205 Invalid end state for parser
    206 Document contains unbalanced elements
    207 Invalid XPath
    208 No such child
    209 Top element does not match start of path
    210 DOM tree unavailable
    302 Can't open file
    401 Invalid XML would be generated
    402 An invalid XML name has been specified

4.3.4.431 @XMLRESET

@XMLRESET[] : Flushes the XML parser buffers, and initializes the parser to its default state.

Returns 0 on success, or an XML error on failure;

XML Errors:

    101 Invalid attribute index
    102 No attributes available
    103 Invalid namespace index
    104 No namespaces available
    105 Invalid element index
    106 No elements available
    107 Attribute does not exist

201 Unbalanced element tag
202 Unknown element prefix (can't find namespace)
203 Unknown attribute prefix (can't find namespace)
204 Invalid XML markup
205 Invalid end state for parser
206 Document contains unbalanced elements
207 Invalid XPath
208 No such child
209 Top element does not match start of path
210 DOM tree unavailable
302 Can't open file
401 Invalid XML would be generated
402 An invalid XML name has been specified

4.3.4.432 @XMLSAVE

@XMLSAVE[*outputfile*] : Saves the modified XML document to a the specified output file.

@XMLSAVE would normally be used when you are creating a document using @XMLINPUT.

Returns 0 on success, or an XML error on failure;

***Example*:**

```
echo %@xmlinput[<test>]
echo %@xmlsave[testfile.json]
echo %@xmlclose[]
```

XML Errors:

101 Invalid attribute index
102 No attributes available
103 Invalid namespace index
104 No namespaces available
105 Invalid element index
106 No elements available
107 Attribute does not exist
201 Unbalanced element tag
202 Unknown element prefix (can't find namespace)
203 Unknown attribute prefix (can't find namespace)
204 Invalid XML markup
205 Invalid end state for parser
206 Document contains unbalanced elements
207 Invalid XPath
208 No such child
209 Top element does not match start of path
210 DOM tree unavailable
302 Can't open file
401 Invalid XML would be generated
402 An invalid XML name has been specified

4.3.4.433 @XMLSTARTELEMENT

@XMLSTARTELEMENT[*name,namespaceURI*] : Writes the opening tag of a new XML element.

Writes the opening tag of a new XML element. If an XML element is already opened, then this element is written as a child.

If *name* is a local name without a prefix, the class will automatically introduce a new xmlns="NamespaceURI" attribute if necessary.

If *name* is in the form prefix:local, then class will automatically introduce a new xmlns:prefix="NamespaceURI" as necessary.

When calling @XMLPutElement or @XMLStartElement, if a NamespaceURI is not specified an empty namespace will be defined for the element. If a namespace should be associated with the element, a NamespaceURI value must be provided. When creating the XML, the class will determine if the namespace already exists to avoid duplicate definitions of the same namespace.

Returns 0 on success, or an XML error on failure;

Example:

To create this XML named *books.xml*:

```
<book xmlns='urn:loc.gov:books' xmlns:isbn='urn:ISBN:0-395-36341-6'>
<title>Cheaper by the Dozen</title>
<isbn:number>1568491379</isbn:number>
</book>
```

Use the code:

```
echo %@xmlcreate[books.xml]
echo %@xmlstartelement[book,urn:loc.gov:books]
echo %@xmlputattr[xmlns:isbn,"",urn:ISBN:0-395-36341-6]
echo %@xmlputelement[title,urn:loc.gov:books,Cheaper by the Dozen]
echo %@xmlputelement[isbn:number,urn:ISBN:0-395-36341-6,1568491379);
echo %@xmlendelement[]
echo %@xmlclose[]
```

XML Errors:

> 101 Invalid attribute index
> 102 No attributes available
> 103 Invalid namespace index
> 104 No namespaces available
> 105 Invalid element index
> 106 No elements available
> 107 Attribute does not exist
> 201 Unbalanced element tag
> 202 Unknown element prefix (can't find namespace)
> 203 Unknown attribute prefix (can't find namespace)
> 204 Invalid XML markup
> 205 Invalid end state for parser
> 206 Document contains unbalanced elements
> 207 Invalid XPath

208 No such child
209 Top element does not match start of path
210 DOM tree unavailable
302 Can't open file
401 Invalid XML would be generated
402 An invalid XML name has been specified

4.3.4.434 @XMLXPATH

@XMLXPATH[[*"filename"],path***]** : XML XPath query. 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.

To return an attribute, preface the attribute name with an @.

The path is a series of one or more element accessors separated by '/'. The path can be absolute (starting with '/') or relative to the current XPath location. The following are possible values for an element accessor:

'name'	A particular element name
name[i]	The i-th subelement of the current element with the given name
[i]	The i-th subelement of the current element
[last()]	The last subelement of the current element
[last()-i]	The subelement located at the last location minus i in the current element
name[@attrname="attrvalue"]	The subelement containing a particular value for a given attribute (supports single AND double quotes)
..	The parent of the current element

Example:

Bookstore.xml :

```
<?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>
```

Bookstore.btm:

```
@ECHO OFF
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[]
```

Running bookstore.btm outputs:

```
 Harry Potter  costs only   29.99
 Learning XML  costs only   39.95
 Day Watch  costs only   14.99
 Winston Churchill: An Autobiography  costs only   49.99
```

XML Errors:

101 Invalid attribute index
102 No attributes available
103 Invalid namespace index
104 No namespaces available
105 Invalid element index
106 No elements available
107 Attribute does not exist
201 Unbalanced element tag
202 Unknown element prefix (can't find namespace)
203 Unknown attribute prefix (can't find namespace)
204 Invalid XML markup
205 Invalid end state for parser
206 Document contains unbalanced elements
207 Invalid XPath
208 No such child
209 Top element does not match start of path
210 DOM tree unavailable
302 Can't open file
401 Invalid XML would be generated
402 An invalid XML name has been specified

4.3.4.435 @YEAR

@YEAR[_date[,format]_**]**: Returns the year for the specified date. See date formats 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)
5 ISO 8601 yyyy-Www-d
6 ISO 8601 yyyy-ddd

Example:

```
echo %@year[5-5-2012,1]
2012
```

4.3.4.436 @YDECODE

@YENCODE[inputfile,outputfile**]** : Encode a file. Returns 0 if the output file was successfully written.

Y Encoding is similar to Base64, but uses 8-bit encoding to reduce the amount of data being sent and received.

Example:

```
echo %@yencode[data.file,data.file.yenc]

echo %@yencode[s,Please encode this string]
```

4.3.4.437 @YENCODE

@YENCODE[inputfile,outputfile**]** : Encode a file. Returns 0 if the output file was successfully written.

Y Encoding is similar to Base64, but uses 8-bit encoding to reduce the amount of data being sent and received.

Example:

echo %@yencode[data.file,data.file.yenc]

4.3.4.438 @ZIPCFILE

@ZIPCFILE[ziparchive,n**]**: Returns the compressed name of file *n* in a .zip archive.

See also [ZIP](#) and [UNZIP](#).

Example:

```
zip tcc *.cpp
echo %@zipcfile[tcc.zip,1]
stdafx.cpp
```

4.3.4.439 @ZIPCFILESIZE

@ZIPCFILESIZE[*ziparchive,n***]**: Returns the compressed size of file *n* in a .zip archive.

See also [ZIP](#) and [UNZIP](#).

Example:

```
zip tcc *.cpp
echo %@zipcfile[tcc.zip,0]
14341
```

4.3.4.440 @ZIPCOMMENT

@ZIPCOMMENT[*ziparchive***]**: Returns the comment (if any) for a .zip archive.

See also [ZIP](#) and [UNZIP](#).

Example:

```
zip /z"TCC source files" tcc *.cpp
echo %@zipcomment[tcc.zip]
TCC source files
```

4.3.4.441 @ZIPCOUNT

@ZIPCOUNT[*ziparchive***]**: Returns the number of files in a .zip archive.

See also [ZIP](#) and [UNZIP](#).

Example:

```
zip /z"TCC source files" tcc *.cpp *.h
echo %@zipcount[tcc.zip]
147
```

4.3.4.442 @ZIPDFILE

@ZIPDFILE[*ziparchive,n***]**: Returns the decompressed name of file *n* in a .zip archive.

See also [ZIP](#) and [UNZIP](#).

Example:

```
zip /R tcc test\
echo %@zipdfile[tcc.zip,4]
test\ntinit.cpp
```

4.3.4.443 @ZIPDFILESIZE

@ZIPDFILESIZE[*ziparchive,n***]**: Returns the decompressed size of file *n* in a .zip archive.

See also [ZIP](#) and [UNZIP](#)

Example:

```
zip tcc *.cpp
echo %@zipdfilesize[tcc.zip,0]
46868.
```

4.3.4.444 @ZIPFILECRC

@ZIPFILECRC[*ziparchive,n***]**: Returns the CRC of file *n* in a .zip archive.

See also ZIP and UNZIP.

Example:

```
zip tcc *.cpp
echo %@zipfilecrc[tcc.zip,0]
244B89DE
```

4.3.4.445 @ZIPFILECOMMENT

@ZIPFILECOMMENT[*ziparchive,n***]**: Returns the comment (description) of file *n* in a .zip archive.

See also ZIP and UNZIP.

Example:

```
describe ntinit.cpp "TCC startup source code"
zip /i tcc *.cpp
echo %@zipfilecomment[tcc.zip,0]
TCC startup source code
```

4.3.4.446 @ZIPFILEDATE

@ZIPFILEDATE[*ziparchive,n***]**: Returns the date and time of file *n* in a .zip archive.

See also ZIP and UNZIP.

Example:

```
zip tcc *.cpp
echo %@zipfiledate[tcc.zip,0]
2021-06-15 13:16:12
```

4.4   Command Line

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-RT and other applications.

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

- Command Names and Parameters
- Conditional Expressions
- Multiple Commands
- Disabling Aliases
- Command Line Length Limits
- Command Grouping

4.4.1 Command Names & Parameters

A TCC-RT command name is 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\tcmd17\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*).

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

4.4.2 Conditional Expressions

The commands DO (when used with the UNTIL or WHILE keyword), IF, IFF/ELSEIFF, and the variable function @IF evaluate a conditional expression, and perform a different action based on whether or not the expression is TRUE. The SWITCH command tests pairs of values for equality. Most of the examples below use the IF 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
▸ status test
▸ logical expression

Relational Expression

A relational expression compares two character strings, using one of the relational operators 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 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. 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* **num2**
> **string1** *relational operator* **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: *expression* is **true** if	string comparison: *expression* is **true if,** when ignoring character case:
EQ or **==**	*num1* equals *num2*	*string1* equals *string2*
NE or **!=**	*num1* does not equal *num2*	*string1* does not equal *string2*
LT	*num1* is less than *num2*	*string1* alphabetically precedes *string2*
LE	*num1* is less than or is equal to *num2*	*string1* alphabetically precedes or is equal to *string2*
GE	*num1* is greater than or is equal to *num2*	*string1* alphabetically succeeds or is equal to *string2*
GT	*num1* is greater than *num2*	*string1* alphabetically succeeds *string2*
EQC	tested as strings ➔	*string1* is identical to *string2*, including character case
=~	regular expression test	*string1* matches the regular expression in *string2*
!~	regular expression test	*string1* doesn't match the regular expression in *string2*

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, TCC-RT recognizes the following additional names for conditions:

CMD	TCC
EQL or **EQU**	**EQ**
NEQ	**NE**
LSS	**LT**
LEQ	**LE**
GTR	**GT**
GEQ	**GE**

Internal variables and variable functions 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-RT 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, array variables, and internal variables do not exist in the environment, so they always fail the DEFINED test.
ERRORLEVEL [relational operator] n	This test retrieves the exit code of the preceding external program. By convention, programs return an exit code of 0 when they are successful and a non-zero number to indicate an error. The relational operator 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 batch files to test the existence of a directory. The expression exist xxx\NUL is true only if xxx is a directory.

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

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

ISBATCH *filename*   If the specified filename is a batch file, the expression is true.

ISDIR *path*
DIREXIST *path*   If the directory specified by path exists, the expression is true. Path may be either absolute or relative. DIREXIST may be used as a synonym for ISDIR.

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

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

ISINTERNAL *command*   If command is an active internal command, the expression is true. Commands can be activated and deactivated with the SETDOS /I command.

ISLABEL *label*   If label 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.

ISLIBRARY *name*   If the name is a library function, the expression is true

ISPLUGIN *name*   If name is a plugin variable, function, or command, the expression is true.

ISREADABLE *filename*   If the filename is readable, the expression is true.

ISSYMLINK *filename*   If the file is a symbolic link, 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.

ISWRITEABLE *filename*   If the filename is writeable, the expression is true.

ISHUNG "title"   If the specified window is not responding, the expression is true.

PLUGIN *module*   If the plugin module is loaded, the expression is true. Do not include an extension (i.e., ".dll"), for the module name.

Logical Expressions

A logical expression is one of the following:

- ▶   a [relational expression](#)
- ▶   a [status test](#)
- ▶   the unary logical operator **NOT** (or **!**) followed by a [logical expression](#)
- ▶   two [logical expression](#)s connected by a binary logical operator

<div align="center">

Logical operators

</div>

operator	type	usage	value is TRUE if
NOT	unary	**NOT** *cond*	*cond* is FALSE.
.AND.	binary	*cond1* **.AND.** *cond2*	both *cond1* and *cond2* are TRUE.
.OR.	binary	*cond1* **.OR.** *cond2*	at least one of *cond1* and *cond2* is TRUE.
.XOR.	binary	*cond1* **.XOR.** *cond2*	one of *cond1* and *cond2* is TRUE, and the other one is FALSE.

This example runs a program called *DATALOAD* if today is Monday or Tuesday (enter this on one line):

```
if "%_dow" == "Mon" .or. "%_dow" == "Tue" dataload
```

Test conditions are always scanned from left to right -- there is no implied order of precedence, as there is in some programming languages. You can, however, force a specific order of testing by grouping conditions with parentheses, for example (enter this on one line):

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

Combining logical expressions

Parentheses can 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
```

4.4.3 Disabling Aliases

If you are not familiar with aliases, see Aliases and the ALIAS command for complete details. See LIBRARY for details on library functions.

At times, you may want to temporarily disable an alias or library function 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.

4.4.4 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 (by default, an ampersand **&**). 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.

You can use multiple commands in alias definitions and batch files as well as from the command line.

4.4.5   Conditional Commands

When an internal command or external program finishes, it returns a result called the exit code. Conditional commands allow you to perform tasks based upon the previous command's exit code. Many programs return 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 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 is non-zero. For example, if the following BACKUP operation fails, then ECHO will display a message:

```
backup c:\ a: || echo Error in the backup!
```

All internal commands return an exit code, but not all external programs do. Conditional commands will behave unpredictably if you use them with external programs which do not return an explicit exit code. To determine whether a particular external program returns a meaningful exit code use an ECHO %? command immediately after the program is finished. If the program's documentation does not discuss exit code, you may need to experiment with a variety of conditions to see how the exit code changes.

4.4.6   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 commands in all subdirectories of your hard disk. You could do it like this:

```
global ren *.wx1 *.wxo
global ren *.tx1 *.txo
```

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

```
global (ren *.wx1 *.wxo & ren *.tx1 *.txo)
```

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

This kind of command grouping is most useful with the EXCEPT, FOR, GLOBAL, and IF 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-RT 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](#) symbols. For example, consider the following batch file fragment which places some header lines (including today's date) and directory displays in an output file using redirection. The first [ECHO](#) command creates the file using >, and the other commands append to the file using >>:

```
echo Data files %_date > filelist
dir *.dat >> filelist
echo. >> filelist
echo Text files %_date >> filelist
dir *.txt >> filelist
```

Using command grouping, these commands can be written much more simply. Enter this example on one line:

```
(echo Data files %_date & dir *.dat & echo `` & echo Text files %_date &
dir *.txt) > filelist
```

The redirection, which appears outside the parentheses, applies to all the commands within the parentheses. Because the redirection is performed only once, the commands will run slightly faster than if each command was entered separately. The same approach can be used for input [redirection](#) and [piping](#).

You can also use command grouping in a batch file or at the prompt to split commands over several lines. This last example is like the redirection example above, but is entered at the prompt. Note the **More?** prompt after each incomplete line. None of the commands are executed until the command group is completed with the closing parenthesis. This example does not have to be entered on one line:

```
[c:\] (echo Data files %_date
More? dir *.dat
More? echo.
More? echo Text files %_date
```

```
More? dir *.txt) > filelist
[c:\]
```

Limitations

A group of commands in parentheses is like a long command line. The total length of the group is only limited by your available RAM.

You cannot use TEXT / ENDTEXT, or GOTO or GOSUB labels in a command group.

Each line you type at the normal prompt or the **More?** prompt, and each individual command within the line, must be within the usual command line length limit.

4.4.7　Starting Applications

TCC-RT offers several ways to start applications.

First, you can simply type the name of any application in a batch file. As long as the application's executable file is in one of the standard search directories (see below), *TCC-RT* 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.

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

TCC-RT has two methods to simplify and speed up access to your applications. One is to create an alias, for example:

```
alias myapp d:\apps\myapp.exe
```

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 or Windows file associations.

Searching for Applications

When you start an application without specifying a path, *TCC-RT* searches for the application in the current directory, and then all directories on the PATH. *TCC-RT* also searches the **Windows** and **Windows system** directories; see the PATH 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-RT* will search each directory for a matching *.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 environment variable. If no such file is found, *TCC-RT* will move on to the next directory in the search sequence.

4.4.8　Escape Character

The escape 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).

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
g	bell (code 7)
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-RT 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 additional usages not related to the above description, as detailed in ASCII, Key Codes and Key Names.

4.4.9 Command Parsing

Whenever you type something at the command line and press the Enter 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-RT goes through several steps when parsing a command line. Before it starts, it writes the entire command line (which may contain multiple commands) to the history log file if history logging has been enabled (with the LOG /H command) 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-RT 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-RT tries to match the command name against its list of aliases. If it finds a match between the command name and one of the aliases you've defined, it replaces the command name with the contents of the alias.

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").

The variable expansion process is modified for certain internal commands, such as EXCEPT, IF, and GLOBAL. These commands are always followed by another command, so variable expansion takes place separately for the original command and the command that follows it.

4.   Identifying a plugin or internal command

Once it has finished variable expansion, TCC-RT next tries to match the resulting command name with its list of plugin commands or internal commands. If it is unsuccessful, it knows that it will have to search for a batch file or external program to execute your command.

You can specify a particular plugin to execute by prefixing the function / variable / command name with the plugin name and a $. For example:

```
echo %_myplugin$variable
echo %@myplugin$func[abc]
myplugin$mycommand
```

5.   Displaying the command

When all of the aliases and environment variables have been expanded, TCC-RT will echo the complete command to the screen (if command line echo has been enabled) and write it to the log file (if command logging has been turned on).

6. Processing redirection and piping

Before it can actually execute your command, TCC-RT must scan the command tail to see if it includes redirection or piping. If so, the proper internal switches are set to send output to an alternate device or to a file instead of to the screen. 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 are processed. However, this might also already have taken place inside some of the variable functions (such as @IF) that are likely to pass escaped strings in their parameters. If you are referencing one of those in an ECHO or similar command, you need to escape twice ("^^") or use SETDOS /X 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-RT will first look for a matching plugin command name; if it doesn't exist then it tries to match an internal command. Otherwise, TCC-RT searches for an executable (*.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).

If the first argument on a command line is in the format "env_var=value command options" (and env_var=value doesn't match an external command) then TCC-RT will set the specified environment variable to the value, execute the command, and then remove the variable.

9. Cleaning up

Once the internal command or external program has terminated, *TCC-RT* 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 command.

4.4.10 Command Line Length Limits

There is no limit to the size of a TCC-RT command line (other than that imposed by Windows or the amount of RAM in the system).

4.4.11 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 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, except for the ISO day format (yyyy-ddd) which requires two fields.

4.4.12 Case Sensitivity

With the following exceptions, TCC-RT treats upper case and lower case letters identically:

The relational operator **EQC** (in IF, IFF, DO, etc.)
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

4.4.13 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-RT has no way of knowing what is a valid argument for external applications.

Directory alias names can be either two or more alphanumeric characters followed by a colon, or a single digit followed by a colon.

Directory aliases support environment variable expansion.

4.5 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, 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
- ▸ Batch Files

4.5.1 Aliases

Much of the power of TCC-RT 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-RT* also supports Directory Aliases, a shorthand way of specifying pathnames. TCC-RT supports either a local alias list that is only visible to the current TCC-RT session, or a global alias list that is shared among all TCC-RT sessions.

This section shows you some examples of the power of aliases. See the ALIAS 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 \.

Aliases can also create customized versions of commands. For example, the DIR command can sort a directory in various ways. You can create an alias called **DE** that means "sort the directory by filename extension, and pause after each page while displaying it" like this:

```
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 command for full 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-RT executes the PUSHD 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 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 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-RT to use the original internal command, not an alias. See Temporarily Disabling Aliases 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 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](#) 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](#) 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](#) 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$** has a special meaning. TCC-RT interprets it to mean "the entire command line, from parameter *n* to the end." If *n* is not specified, it has a default value of **1**, so **%$** 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 and UNALIAS commands for more information and examples.

4.5.2   Batch Files

A batch file is a file that contains a list of commands to execute. TCC-RT reads and interprets each line as if it had been typed at the keyboard. Like aliases, batch files are handy for automating computing tasks. Unlike aliases, batch files can be as long as you wish. Batch files take up separate disk space for each file, and can't usually execute quite as quickly as aliases, since they must be read from the disk.

Some of the topics included in this section are:

- .BAT, .CMD, and .BTM
- Echoing in Batch Files
- Batch File Line Continuation
- Batch File Parameters
- Using Environment Variables
- Batch File Commands
- Interrupting a Batch File
- Automatic Batch Files
- Detecting TCC and Take Command
- Using Aliases in Batch Files
- String Processing
- Batch File
- Lua Support
- Perl Support
- Perl Support
- Python Support
- REXX Support
- Tcl/tk Support
- EXTPROC / Shebang Support

4.5.2.1   .BAT, .CMD & .BTM Files

A batch file can run in two different modes. In the first, traditional mode, each line of the batch file is read and executed individually, and the file is opened and closed to read each line. In the second mode the batch file is opened once, the entire file is read into memory, and the file is closed. Only the first mode 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 command.

4.5.2.2   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 OFF and ECHO ON commands.

▸   The default setting can be changed with the SETDOS /V command.

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-RT also has a command line echo that is unrelated to the batch file echo setting. See ECHO for details about both settings.

4.5.2.3   Special syntax for CMD compatibility

For compatibility with CMD, TCC-RT supports additional syntax to qualify references to parameters of batch files and the control variable of the FOR command when referenced by the *command* it executes. However, this syntax can usually be replaced by more flexible Variable Functions.

CMD 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	`%@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.

%~$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.

4.5.2.4 Batch File Line Continuation

TCC will combine multiple lines in the batch file into a single line for processing when the escape character 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
```

4.5.2.5 Batch File Parameters

Like aliases, user-defined functions 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, commas, and equals signs (=) 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 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
`%n$`	the command tail starting with parameter number **n**, modified by SHIFT
`%-n$`	the command tail from parameter 1 to **n** - 1
`%$`	the complete command tail, modified by SHIFT
`%*`	the complete command tail, unmodified by SHIFT

For example, **%3$** means the third and all subsequent parameters. The values of **%#, %n$, %-n$,** and **%$** will change if you use the SHIFT command. To emulate CMD, SHIFT 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 and IFF commands.)

Batch files can also use environment variables, internal variables, and variable functions.

Batch file parameters may also use the special CMD compatibility syntax.

4.5.2.5.1 Parameter Quoting

As TCC-RT parses the command line, it looks for the command separator, conditional commands (| | and &&), white space (spaces, tabs, and commas), percent signs % which indicate variables or batch file parameters to be expanded, and redirection and piping 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-RT 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 for details.

You can disable and reenable back quotes and double quotes with the SETDOS /X command.

4.5.2.6 Using Environment Variables

Batch files can use environment variables, internal variables, variable functions, or user-defined functions. 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, use the SET command to store the information you want in an environment variable. Pick a variable name that isn't likely to be in use by some other program (for example, PATH would be a bad choice), and use the UNSET command to remove these variables from the environment at the end of your batch file. You can use SETLOCAL and ENDLOCAL to create a "local" environment so that the original environment will be restored when your batch file is finished.

Environment variables used in a batch file may contain either numbers or text. It is up to you to keep track of what's in each variable and use it appropriately; if you don't (for example, if you use %@EVAL to add a number to a text string), you'll get an error message or a meaningless return value.

4.5.2.7 Batch File Commands

Some commands are particularly suited to batch file processing. Each command is explained in detail in the Command Reference. Here is a list of some of the commands you might find most useful:

ACTIVATE	activates another window
BEEP	produces a sound of any pitch and duration through the computer's speaker
BREAKPOINT	set a breakpoint in the batch debugger
CALL	executes one batch file from within another
CANCEL	terminates all batch file processing
CLS	clears the TCC-RT window
COLOR	sets the TCC-RT display colors
DEBUGSTRING	send text to the debugger
DEFER	defers a command until the batch file ends
DO	starts a loop. The loop can be based on a counter, or on a conditional expression, strings, or files. ENDDO terminates the loop
DRAWBOX	draws a box on the screen
DRAWHLINE	draws horizontal lines on the screen
DRAWVLINE	draws vertical lines on the screen
ECHO	sends text to the standard output device
ECHOS	sends text to the standard output device
ECHOERR	sends text to the standard error device
ECHOSERR	sends text to the standard error device
ENDLOCAL	restores the settings that were saved and allows specific variables to be exported (see SETLOCAL)
ENDTEXT	ends the block of text started with TEXT
EVENTLOG	writes a string to the Windows application event log
FOR	executes commands for each file that matches a set of wildcards, or each entry in a list
GOSUB	executes a subroutine inside a batch file (see RETURN).
GOTO	branches to a different location in the batch file
IF	execute commands based on a conditional expression
IFF	
INKEY	collects keyboard input and store it in environment variables
INPUT	collects keyboard input and store it in environment variables
JABBER	send an instant message (IM)
KEYSTACK	sends keystrokes to applications
LOADBTM	changes the batch file operating mode
LOCAL	define variables local to a library function or batch file
MSGBOX	displays a dialog box with standard buttons like Yes, No, OK, and Cancel, and returns the user's selection
ON	initializes error handling for Ctrl-C / Ctrl-Break, or for program and command errors
OSD	Display floating text on the desktop
PAUSE	displays a message and waits for the user to press a key
PDIR	creates a customized DIR-like display of directory contents
PLAYAVI	plays Windows .AVI files
PLAYSOUND	plays Windows sound files
POSTMSG	send a message to a window
QUERYBOX	displays a dialog box for text input
QUIT	ends the current batch file and optionally returns an exit code

[REM](#)	places a remark in a batch file
[RETURN](#)	terminates a subroutine (see [GOSUB](#))
[SCREEN](#)	positions the cursor on the screen and optionally prints a message at the new location
[SCRPUT](#)	displays a message in color
[SENDMAIL](#)	sends an email message
[SETLOCAL](#)	saves the current disk drive, default directory, environment, alias list, and special character settings (see [ENDLOCAL](#)).
[SHIFT](#)	changes the numbering of the batch file parameters
[SMPP](#)	sends messages using the SMPP protocol
[SNPP](#)	sends a message to an alphanumeric pager
[START](#)	starts another session or window
[SWITCH](#)	selects a group of statements to execute based on the value of a variable
[TEXT](#)	displays a block of text (see [ENDTEXT](#))
[TIMER](#)	starts or reads a stopwatch
[TITLE](#)	changes the window title
[VSCRPUT](#)	displays a vertical message in color
[WMIQUERY](#)	Query the Windows Management Instrumentation interface
[WMIRUN](#)	Run WMI methods on local or remote machines

These commands, along with the internal variables and variable functions, make the enhanced batch file language extremely powerful.

4.5.2.8 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-RT or an application program is running, how the application, if any, was written, whether [BREAK](#) is ON or OFF, and whether the [ON BREAK](#) command is in use.

If TCC-RT 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](#) command; answering **A** is similar to the [CANCEL](#) command.

4.5.2.9 Detecting TCC and Take Command

From a batch file, you can determine if TCC-RT is loaded by doing a numeric comparison:

```
if 01 == 1 echo TCC-RT is loaded!
```

In *TCC-RT*, this is a numeric comparison and true; in CMD it is a string comparison and false. Once you have established that the batch file is running in *TCC*, you can use internal variables like _CMDPROC, _4VER, _DOS, _DOSVER, and _WIN to further determine the operating environment.

You can determine if *TCC* is running in a *Take Command* tab window with the internal variable [_TCTAB](#):

```
if %_tctab == 1 echo TCC is running in a Take Command tab window!
```

You can prevent your batch file from running in CMD by giving it the .BTM extension. CMD doesn't recognize .BTM files as batch files.

4.5.2.10 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 *
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 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 command. The rest of the command line is interpreted as another command plus possible command line parameters, which the alias executes. This alias is used here to switch to a directory, run an application, and switch back. It could also be used from the command line.

The following 9 lines print a menu on the screen and then get a keystroke from the user and store the keystroke in an environment variable called *userchoice*. Then the SWITCH command is used to test the user's keystroke and to decide what action to take.

There's another side to aliases in batch files. If you're going to distribute your batch files to others, you need to remember that they may have aliases defined for the commands you're going to use. For example, if the user has aliased CD to CDD and you aren't expecting this, your file may not work as you intended. There are two ways to address this problem.

The simplest method is to use SETLOCAL, ENDLOCAL, and UNALIAS to clear out aliases before your batch file starts, and SETDOS to select the special characters you depend on, and restore them at the end, as we did in the previous example. Remember that SETLOCAL and ENDLOCAL will save and restore not only the aliases but also the environment, the current drive and directory, and various special characters.

If this method isn't appropriate or necessary for the batch file you're working on, you can also use an asterisk * before the name of any command. The asterisk means the command that follows it should not be interpreted as an alias. For example the following command redirects a list of file names to the file *FILELIST*:

```
dir /b > filelist
```

However, if the user has redefined DIR with an alias this command may not do what you want. To get around this just use:

```
*dir /b > filelist
```

The same can be done for any command in your batch file. If you use the asterisk, it will disable alias processing, and the rest of the command will be processed normally as an internal command, external command, or batch file. Using an asterisk before a command will work whether or not there is actually an alias defined with the same name as the command. If there is no alias with that name, the asterisk will be ignored and the command will be processed as if the asterisk wasn't there.

4.5.2.11 String Processing

As you gain experience with batch files, you're likely to find that you need to manipulate text strings. You may need to prompt a user for a name or password, process a list of files, or find a name in a phone list. All of these are examples of string processing -- the manipulation of readable text.

TCC-RT includes several features that make string processing easier. For example, you can use the INPUT, MSGBOX, and QUERYBOX commands for user input; the ECHO and ECHOERR, ECHOS and ECHOSERR, SCREEN, SCRPUT, and VSCRPUT commands for output; and the FOR command or the @FILEREAD function to scan through the lines of a file. In addition, variable functions offer a wide range of strings and character handling 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 command at the beginning of this batch file saves the environment and aliases. Then the UNALIAS * command removes any existing aliases so they won't interfere with the behavior of the commands in the remainder of the batch file. The first block of lines ends with a INPUT command which asks the user to enter a name. The user's input is stored in the environment variable NAME.

The second block of lines extracts the user's first and last names from the NAME variable and calculates the length of each. It stores the first and last name, along with the length of each, in additional environment variables. Note that the @WORD function numbers the first word as 0, not as 1.

The IFF command in the third block of lines tests the length of both the first and last names. If either is longer than 8 characters, the batch file displays an error message and ends. (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 command to restore the original environment and alias list. At the same time, ENDLOCAL discards the temporary variables that the batch file used (NAME, FIRST, FLEN, etc.).

When you're processing strings, you also need to avoid some common traps. The biggest one is handling special characters.

Suppose you have a batch file with these two commands, which simply accept a string and display it:

```
input Enter a string:  %%str
echo %str
```

Those lines look safe, but what happens if the user enters the string "some > none" (without the quotes). After the string is placed in the variable STR, the second line becomes

```
echo some > none
```

The ">" is a [redirection](#) symbol, so the line echoes the string "some" and redirects it to a file called NONE -- probably not what you expected. You could try using [double quotes](#) 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](#) 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 [Lua](#), [Perl](#), [Python](#), or [REXX](#) languages. Our products can execute Perl, Python, REXX, and Ruby programs internally, and also support evaluating individual Perl, Python, REXX, and Ruby expressions internally.

4.5.2.12 Batch File Compression

If you have **Take Command** or **TCC**, you can compress your *.BTM* files for use with **TCC-RT** with the internal BATCOMP command. That command reduces the size of large batch files by at least a half and makes them unreadable with the [LIST](#) 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 [/Q][/O] InputFile OutputFile
```

You must specify the full name of the input file and output files, including their extensions, on the BATCOMP command line. For example, to compress *MYBATCH.CMD* and save the result as *MYBATCH.BTM*, you use this command:

```
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.

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.

4.5.2.13   Lua support

Lua is a powerful, fast, lightweight, embeddable scripting language. *TCC-RT* includes internal support for Lua, both for executing Lua scripts and for executing individual Lua expressions. The version supplied with **TCC-RT** is Lua 5.4.6. For more information on Lua, go to http://www.lua.org.

You must enable Lua support in the OPTION / Startup page. If it is enabled, TCC-RT will automatically load LUA.DLL on your system. TCC-RT checks to see if you are running a *.LUA* file. If so, TCC-RT passes the file to the Lua interpreter for processing.

See also: the @LUA function.

4.5.2.14   Perl support

Perl is a powerful file and text processing language available on many platforms. Perl is a useful extension to the TCC-RT 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-RT** is PerlScript (the WSH COM interface), which is included in Active State Perl (free from www.activestate.com).

You must enable Perl support in the OPTION / Startup page. If it is enabled, TCC-RT will automatically load Perl on your system. If a suitable library is found, TCC-RT checks to see if you are running a *.PL* file. If so, TCC-RT passes the file to your Perl interpreter for processing.

See also: the @PERL, @PYTHON, @REXX and @RUBY functions.

4.5.2.15   Python support

Python is a powerful file and text processing language available on many platforms. Python is an ideal extension to the TCC-RT batch language, especially if you need advanced string processing capabilities.

The Python language is not built into **TCC-RT**, and must be obtained separately. The version supported by **TCC-RT is** from https://python.org. TCC-RT supports versions 3.12, 3.11, 3.10, 3.9, 3.8, 3.7, 3.6, 3.5, 3.4, 3.3, 3.2, 3.1, 2.7, 2.6, and 2.5. (TCC-RT will search for the Python dll's in that order.)

You must enable Python support in the OPTION / Startup page. If it is enabled, TCC-RT will automatically load a Python interpreter when it starts. If a suitable library is found, TCC-RT checks to see if you are running a *.PY* file. If so, TCC-RT passes the file to your Python interpreter for processing.

See also: the @PYTHON, @LUA, @PERL, @REXX and @RUBY functions.

4.5.2.16 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-RT batch language, especially if you need advanced string processing capabilities.

The REXX language is not built into **TCC**, and must be obtained separately by downloading the free Regina REXX from http://regina-rexx.sourceforge.net/, or ooREXX (Open Object REXX) from http://www.oorexx.org/.

You must enable REXX support in the OPTION / Startup page. If it is enabled, when TCC-RT loads it asks Windows to locate specific REXX libraries associated with Regina or Open Object REXX. If a REXX library is found, TCC-RT 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-RT passes the file to your REXX interpreter for processing.

When you send a command from a REXX program back to TCC-RT 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-RT. TCC-RT uses the address **CMD** for compatibility with scripts written for CMD.

For details on communication between REXX and **TCC**, or for more information on any aspect of REXX, see the Regina or ooREXX documentation.

See also: the @REXX, @PERL, @PYTHON, and @RUBY functions.

4.5.2.17 Tcl/tk Support

The Tcl/tk language is not built into **TCC-RT**, and must be obtained separately. The version supported by **TCC-RT** is ActiveTcl 8.6 (free from www.activestate.com).

You must enable Tcl support in the OPTION / Startup page. If it is enabled, TCC-RT will automatically load a Tcl interpreter when it starts. If a suitable library is found, TCC-RT checks to see if you are running a *.TCL* file. If so, TCC-RT passes the file to your Tcl interpreter for processing.

It's not possible for TCC-RT to determine in advance whether you're running a Tcl or a Tk script. After executing the script, **TCC-RT** checks if a Tk window is running. If so, it enters a Tk event loop and waits for the window to be closed. If not, TCC-RT assumes it was a Tcl script and TCC-RT returns immediately.

Because of the way the Tk interpreter works, it is not possible for TCC-RT to maintain a persistent interpreter after executing a Tk script. TCC-RT will close the current Tcl/tk interpreter and create a new one the next time a Tcl / tk script is executed.

See also @TCL and @TK.

4.5.2.18 EXTPROC / SHEBANG Support

TCC-RT 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-RT 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-RT 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-RT 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-RT 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*.

4.6   File Selection

Most internal commands (like COPY, DIR, 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-RT 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
- ▸ Executable Extensions
- ▸ Using Internet URLs
- ▸ Using FTP and HTTP Servers
- ▸ OpenAFS
- ▸ Ranges
- ▸ Attribute Switches
- ▸ Multiple Filenames
- ▸ Include Lists
- ▸ Delayed Variable Expansion
- ▸ Extended Parent Directory Names
- ▸ LFN File Searches
- ▸ @File Lists
- ▸ Command Switches for File Selection

4.6.1 Wildcards and Regular Expressions

Wildcards let you specify a file or group of files by typing a partial filename. The appropriate directory is scanned to find all of the files that match the partial name. You can also specify files with regular expressions.

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 (for internal commands), or EXCEPT (for external commands)**.**

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

WARNING: When you use a wildcard search for files to process in a command like FOR or DO, 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.

TCC-RT also supports wildcards in the directory names (but not in the drive name). You can control the subdirectory recursion by specifying * or ** in the path. A * will match a single subdirectory level; a ** will match any all subdirectory levels for that pathname. Directory wildcards also support regular expressions. Directory wildcards cannot be used with the /O:... option (which sorts entries before executing the command). And think very carefully before using directory wildcards with a /S (recurse subdirectories) option, as this will almost certainly return unexpected results! There are a few commands which do not support directory wildcards, as they would be meaningless or destructive (for example, TREE, @FILEOPEN, @FILEDATE, etc.).

For example:

`del c:\test\test2\*\foobar`	Delete the file **foobar** in any subdirectory of **c:\test\test2** (but not in any of their subdirectories).
`del c:\test\**\*foo*\foobar`	Delete the file **foobar** in any subdirectory under **c:\test** (and all of their subdirectories) that has "foo" anywhere in the name.
`del c:\test\t*2\foobar`	Delete the file **foobar** in any subdirectory of **c:\test** that begins with a **t** and ends with a **2.**

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-RT 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-RT.

"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-RT 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*.

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. 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-RT** will attempt to delete *LETTER1* and then *V2*, because the semicolon indicates an include list. 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 configuration option is set, wildcard searches accept a match on either the LFN or the SFN to match the behavior of CMD. This may cause some files to be found 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-style "extension" matching (only one extension, at the end of the word) and the advanced TCC-RT filename matching (allowing things like *.*.abc) when an asterisk is encountered in the destination of a COPY, MOVE or REN/RENAME 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 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.

Take Command and *TCC* include a regular expression analyzer dialog (Ctrl-F7 from the *TCC* command line, or under the Tools menu in *Take Command*.) There are two edit boxes:

1) The first is for the regular expression to test.. If the regular expression is valid, the dialog will display a green check to the right of the expression edit box. If the regular expression is invalid, the dialog will display a red X.
2) The second edit box is for the text you want to match against the regular expression. If the text matches the regex, the dialog will display a green check to the right of the test edit box. If the text doesn't match, the dialog will display a red X.

You can choose the regular expression syntax you want to use (Perl, Ruby, Java, etc.). The analyzer will default to the current default for *Take Command* and *TCC*.

Regular Expression Analyzer					✕
Expression:					
Test:					
Regular expression syntax	.	Any character	\s+	Whitespace	
○ PERL	[]	Character in range	[[:alnum:]]	Alphanumeric character	
○ Ruby	[^]	Character not in range	[[:alpha:]]	Alphabetic character	
○ Java	^	Beginning of line	\d	Decimal digit	
○ grep	$	End of line	[-+]?\d+	Integer number	
○ POSIX	|	Or	[\da-fA-F]+	Hex number	
○ GNU	?	Optional	[[:alpha:]]+	Alphabetic string	
○ Python	*	0 or more matches	"([^"\\]\\.)*"	Quoted string	
○ Emacs	+	1 or more matches	{n,m}	Range	

4.6.2   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-RT looks for a file with that name to execute.

The file's extension may be *.EXE* to indicate that it contains a program; *.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.

You can add to the default list of extensions, and have TCC-RT 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 for details on this feature and its relationship to executable extensions. See also: Executable Files and File Searches.

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 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-RT will search the PATH for it if necessary. However, you can make sure that the correct program or batch file is used, and speed up the executable extension, by specifying the full name including drive, path, filename, and extension. You can utilize other environment variables in the specification.

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

The next example defines *WORDPAD.EXE* (a Windows editor) as the processor for *.TXT* files:

```
set .txt="c:\program files\accessories\wordpad.exe"
```

Now, if you have a file called *HELLO.TXT* and enter the command

```
hello
```

TCC-RT 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-RT would execute the command:

```
"c:\program files\accessories\wordpad.exe" c:\source\hello.txt -w
```

In order for executable extensions to work, the command, program, batch file, or alias must be able to interpret the command line properly. For example, if a program you want to run doesn't accept a file name on its command line as shown in these examples, then executable extensions won't work with that program.

Executable extensions may include wildcards, so you could, for example, run your text editor for any file with an extension beginning with *T* by defining an executable extension called *.T*. Extended wildcards (*e.g.*, **DO[CT]** for *.DOC* and *.DOT* files) may also be used.

To remove an executable extension, use UNSET to remove the corresponding variable.

4.6.3   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.

HTTP and HTTPS addresses in *Take Command* and *TCC* will have any embedded spaces converted to "%20" before sending the URL to the server.

4.6.4   Using FTP and HTTP Servers

TCC-RT allows direct access to remote servers from internal commands such as COPY, DEL, DIR, MOVE, MD, RD, REN, and SELECT via several protocols:

- ▸ FTP (basic FTP)
- ▸ TFTP (Trivial FTP)
- ▸ FTPS (SSL FTP)
- ▸ SFTP (SSH FTP)
- ▸ HTTP (basic Web access)
- ▸ HTTPS (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 Microsoft FTP site, you could use this command:

```
dir ftp://ftp.microsoft.com/*
```

If you don't specify a username and password, TCC-RT 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 configuration option. You must add entries to the *FTP.CFG* file manually. The format for each line is:

> ***url [(alias)] username password [directory template]***

For example:

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

You can have multiple users for a single FTP site (for example, an admin user and and a normal user). You need to add an alias (enclosed in parentheses) following the name of the ftp site. For example:

```
ftp://ftp.jpsoft.com (jpadmin) Bob AdminPassword
ftp://ftp.jpsoft.com (jppublic) anonymous Bob@ftp.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-RT 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 specify a password of *****, you will be prompted to enter the password (which will appear on the screen as asterisks). Depending on the type of operation you're doing, you may need to enter the password multiple times as TCC-RT repeatedly connects and disconnects from the server. To avoid this, use IFTP (which will also be much faster).

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 command to start an FTP session with a server and then use a simplified syntax to manipulate files on the server.

TCC-RT also supports symbolic hostnames (defined in \windows\system32\drivers\etc\hosts).

TCC-RT 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@ftp.microsoft.com/
```

TCC-RT 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-RT 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-RT will instead pass the exact string directly to the remote server.

TCC-RT 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-RT 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. (If you include wildcards in the filename, TCC-RT will match filenames regardless of case.) We urge you to test each server you use with nondestructive commands like DIR before you try to copy or delete files, create or remove directories, etc.

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

Note: If you use a partial reference such as *ftp:mydir* outside the scope of an IFTP command, TCC-RT 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 connection. You can determine if there is an active IFTP connection with the _iftp and _iftps variables.

Before you can use the built-in FTP support or the IFTP command, you must establish the necessary connection to the Internet. For example, if you use Windows Dial-Up Networking to connect to the Internet, you must start your dial up connection first. If you connect through a proxy server, you must set the Proxy configuration options.

TCC-RT will try to preserve timestamps when transferring files. The MDTM command is used when downloading, and the MFTM command is used when uploading. If the FTP server doesn't support these commands, TCC-RT will use the current date/time for the timestamp.

Non-standard FTP servers:

TCC-RT 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-RT 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:

l"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 ! ! ! Z T D U F
```

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

See the FTP section above for general notes and requirements.

TFTP is only available with COPY (and with MOVE 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 section above for general notes and requirements.

The **HTTP** syntax is:

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

For example:

```
copy http://jpsoft.com/downloads/v28/tcmd.exe
```

TCC-RT supports HTTP compression when receiving data.

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

See the FTP section above for general notes and requirements.

The **FTPS** syntax is:

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

For example:

```
copy ftps://bob:pass@ftp.myserver.com/tcmd/tcmd.exe
```

TCC-RT will detect if it is running detached or as a service before prompting for SSL authentication, and will provide an automatic **Y** (yes) input.

- **SFTP ("SSH FTP") support:**

See the FTP section above for general notes and requirements.

The S**FTP** syntax is:

```
sftp://[user:password@]server[:port]/filename
```

For example:

```
copy sftp://bob:pass@ftp.myserver.com/tcmd/tcmd.exe
```

TCC-RT will detect if it is running detached or as a service before prompting for SSH authentication, and will provide an automatic **Y** (yes) input.

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

See the FTP section above for general syntax and requirements.

The **HTTPS** syntax is:

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

For example:

```
copy https://jpsoft.com/downloads/v28/tcmd.exe
```

TCC-RT supports HTTP compression when receiving data.

4.6.5   OpenAFS

TCC-RT 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.

4.6.6   Ranges

Most internal commands which accept wild cards also allow size, date, time, exclusion, description, and owner ranges to further define the files that you wish to work with. TCC-RT 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 **/[s100,200]**.

A description range begins with **/l**. See Description Ranges for the full syntax.

If you use the syntax **/[=]**, TCC will display a dialog that allows you to select the ranges you want. For example:

copy /[=] file1 file2

The Date, Time, Size, Owner, and Description ranges support the ! (NOT) operator to reverse the test.

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,

```
/[d2018-2-8,2019-2-9] /[s1024,2048]
```

means files last modified between February 8, 2018 and February 9, 2019, 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, 2018, you could use this command:

```
dir /[d2018-10-1,+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 or _TIME) and built-in variable functions (*e.g.* @DATE, @TIME, @MAKEDATE, @MAKETIME, @FILEDATE, @FILETIME, or @EVAL). 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
- Date Ranges
- Time Ranges
- Exclusion Ranges
- Owner Ranges
- Description Ranges

Ranges can be used with many commands, including ATTRIB, COPY, DEL, DESCRIBE, DIR, DO, EXCEPT, FFIND, FOR, HEAD, LIST, MOVE, PDIR, RD, REN, SELECT, TAIL, and TYPE

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 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 2018-10-27 and 2010-18-30 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-RT 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]**.

4.6.6.1 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 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
p	1,000,000,000,000,000	10**15	P	1,125,899,906,842,624	2**50	petabyte

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

4.6.6.2   Date Ranges

Date ranges select files dated at any time of day between the inclusive limits specified. For example, **/ [d2011-12-1,2011-12-5]** selects files that were last modified on *or* after December 1, 2011, but not modified *after* December 5, 2011.

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 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, **/[d2010-12-31,2011-1-1]** selects files modified between December 31, 2010 and January 1, 2011.

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 **/[d2010-7-01@8:00a,2010-7-03@6:00p]** selects files that were modified at any time between 8:00:00 am on July 1, 2010 and 6:00:00 pm on July 3, 2010. 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-RT substitutes the current date and time. For example, **/[d2010-10-1]** selects files dated between October 1, 2010 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
/[d2010-1-27,+3]	modified between January 27, 2010 and January 30, 2010
/[d2010-1-27,-3]	modified between January 24, 2010 and January 27, 2010
/[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.

Instead of a date, you can specify a file age for the first and/or second parameter. See Time Stamps, @AGEDATE and @MAKEAGE.

To select files last modified *n* days ago or earlier, use **/[d-n,1980-1-1]**. 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-3,1980-1-1]
```

This reversed date range (with the later date given first) will be handled correctly by TCC-RT. 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 for FAT / VFAT, or January 1, 1601 for NTFS.

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, **/[d2010-12-08@12:00,+2@12:00]** selects files that were last modified between noon on December 8 and noon on December 10, 2010. 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**.**
- 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, 2010 and February 7, 2010, inclusive, you would use **/[dc2010-02-**

1,2010-2-7]. If you don't specify which date and time to use, TCC-RT 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)

4.6.6.3   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-RT 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 for additional details.

Defaults

Start time:	Current time
End time:	Current time
Time type:	Modification (last write)

4.6.6.4   File Exclusion

Most internal commands which accept wildcards also accept file exclusion ranges to further define the files that you wish to work with. TCC-RT 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 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 **]**. You can specify multiple file exclusions (useful if you have a alias that is defining an exclusion and you want to pass another one as an argument).

Inside the brackets, you can list one or more filenames to be excluded from the command. The filenames can include wildcards and extended wildcards, but 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. 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 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 or FOR 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.

4.6.6.5   Owner Ranges

Most internal commands which accept wildcards also accept owner ranges to further define the files that you wish to work with. TCC-RT examines each file or directory and excludes those whose owner doesn't match that in the exclusion range.

Owner ranges support wildcard comparisons. The value is the same as shown in DIR /Q or %@owner.

The syntax is:

```
/[O"owner"]
```

If you precede the **O** with a **!**, the result is reversed.

The following example will display all files in the current directory owned by Bob:

```
dir /[O"*\Bob"] *
```

The following example will display all files in the current directory **except** those owned by Bob:

```
dir /[!O"*\Bob"] *
```

4.6.6.6   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 for additional details.

A description range is specified as **/I"*text*"** where *text* is the description to be matched. Wildcards are supported. For example, **/I"*agua*"** selects all files with the string **agua** somewhere in the file description. The search text must be enclosed in double quotes, and must immediately follow the **/I**, with no intervening spaces.

You can select all files that have a description with **/I"[?]*"** (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 **/I"[]"**   (the **[]** requires that the first character, and therefore the descriptor itself, does not exist).

You can also search descriptions using regular expressions with **/R"*text*".**

If you precede the **I** or **R** with a **!**, the result is reversed. For example, **/!I"*beta*"** will select all of the files that do **not** have the word *beta* in their description.

See DESCRIBE for details on file descriptions.

4.6.7   Attribute Switches

Most file commands in TCC-RT include the **/A:** switch, which allows you to select files for the command to process based on their attributes. These switches all use the format **/A[:][-+]RHSAD**. The colon after /A is optional in DIR, FFIND, and SELECT, but is required in all other 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 volumes, the extended attributes below are also available.

E	Encrypted
C	Compressed
P	Sparse file
I	Not content-indexed
L	Symbolic link or Junction (reparse point)
N	Normal (cannot be used for file selection)
O	Offline
P	Pinned (Windows 10 OneDrive)
T	Temporary
U	Unpinned (Windows 10 OneDrive)
V	Integrity (Windows Server 2012R2+ ReFS only)
X	No scrub data (Windows Server 2012R2+ ReFS only)

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.

If you use the format **/A:=**, TCC will display a dialog that allows you to select the attributes you want:
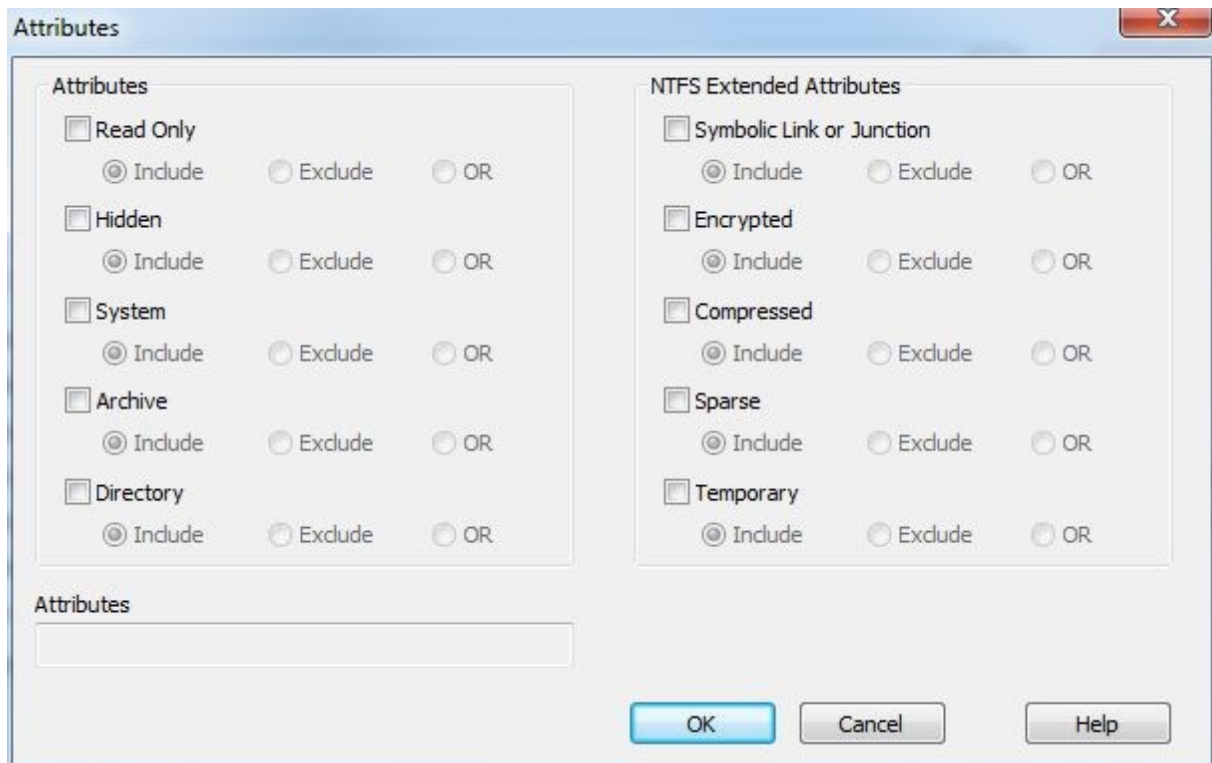
Example

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
```

4.6.8 Multiple Filenames

Most file processing commands can work with multiple files at one time. To use multiple file names, you simply list the files one after another on the command line, separated by spaces. You can use wildcards in any or all of the filenames. For example, to copy all `.TXT` and `.DOC` files from the current directory to drive **A,** you could use this command:

```
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 or MOVE, be sure that you always include a specific destination on the command line. If you

don't, the command will assume that the last filename is the destination and may overwrite important files.

Like extended wildcards and include lists, 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 as well as the DO, FOR and SELECT commands for other ways of passing multiple file names to a command.

4.6.9   Include Lists

Any internal command that accepts multiple filenames will also accept one or more include lists. An include list is simply a group of filenames, with or without wildcards, separated by semicolons [;]. 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 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, because the last name in the list will be part of the include list, and won't be seen as the destination file name. Include lists can only be used as the source parameter -- the location files are coming from -- for COPY and other similar commands. They cannot be used to specify a destination for files.

- Third, multiple filenames and include lists are processed differently by the DIR and SELECT commands. If you use multiple filenames, all of the files matching the first filename are processed, then all of the files matching the second name, and so on. When you use an include list, all files that match any entry in the include list are processed together, and will appear together in the directory display or SELECT list. You can see this difference clearly if you experiment with both techniques and the DIR command. For example,

  ```
  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](#) and [multiple filenames](#), include lists work with internal commands, but not with external programs (unless they have been programmed especially to support them).

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-RT to treat it as an include list.

If a filename parameter includes a semicolon, TCC-RT 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](#).

4.6.10 @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](#) *@XXX* specifies that [LIST](#) 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 containing the names of the files to process, one per line. This allows you to create a list of files for processing using output from [DIR](#) /B, [DIR](#) /F, or [FFIND](#), 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](#)	[FOR](#)	[TAIL](#)
[COPY](#)	[HEAD](#)	[TOUCH](#)
[DEL / ERASE](#)	[LIST](#)	[TYPE](#)
[DESCRIBE](#)	[MOVE](#)	[ZIP](#)
[DO](#)	[RD / RMDIR](#)	
[EXCEPT](#)	[REN / RENAME](#)	

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.

4.6.11 Delayed Variable Expansion

Some of the internal commands (COPY, MOVE, PDIR, REN) 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
```

4.6.12 Extended Parent Directory Names

TCC-RT 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 function to expand file and directory references when necessary:

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

4.6.13 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 (LFN) specified, and an SFN automatically generated by the file system. The SFN associated with an LFN may change when the file is moved or copied even when the LFN is not changed.

When CMD 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
Letter Home.DOC	*LETTER~1.DOC*
Letter02.DOC	*LETTER02.DOC*

A search for *LETTER??.DOC* will find both files. The second file (*Letter02.DOC*) will be found during the search of long filenames. The first file (*Letter Home.DOC*) will be found during the search of short filenames but will return LFN.

Because this dual search can result in some very unexpected or even disastrous results, **TCC-RT** 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!

4.6.14 Switches for File Selection

Many of the file processing commands ([ATTRIB](#), [COPY](#), [DEL](#), [DESCRIBE](#), [HEAD](#), [MOVE](#), [REN](#), [TAIL](#), [TYPE](#), etc.) support several standard switches for selecting files to process. Be sure to see the individual commands for details on which switches are supported for each command and how they work, and for additional switches specific to each command. Make sure that any [range](#) selections precede the options below in the command line.

The common file selection switches include:

`/A:[[-+] rhsadecijlopt]`	Select files based on their attributes, for example **/A:RH** selects files which have the read-only and hidden attributes set. See [Attribute Switches](#) for details; see [File Attributes](#) 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.

4.7   Input / Output Redirection

This section covers features to change how **TCC-RT** 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-RT**, 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 command, the task is easy. DIR's output goes to the standard output device, and you have redirected standard output to the printer, so the DIR command prints filenames instead of displaying them on the screen. You can just as easily send the output of DIR (or any other command) to a file or a serial port.

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

Redirection and piping affect the standard input, standard output, and standard error devices. They do not work with application programs which read the keyboard hardware directly, or which write directly to the 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 and Y commands are "pipe fittings" which add more flexibility to pipes.

TCC-RT's output is normally in ANSI. If you want to redirect output in Unicode, you need to either use the /U startup option in *TCC-RT*, or the Unicode Output option in TCMD.INI.

- Redirection and Piping
- Keystack
- Page and File Prompts

4.7.1   Redirection and Pipes

This section covers redirection and pipes. You can use these features to change how TCC-RT 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 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.

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 PEE, TEE and Y commands are "pipe fittings" which add more flexibility to pipes.

TCC's output is normally in ANSI. If you want to redirect output in Unicode, you need to either use the /U startup option in *TCC*, or the Unicode Output option in TCMD.INI.

4.7.1.1 Redirection

Redirection can be used to reassign the standard input (stdin), standard output (stdout), and standard error (stderr) devices from their default settings (the keyboard and screen) to another device such as NUL or serial port, to a file, or to the Windows clipboard. You must use some discretion when you use redirection with a device.

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.

TCC's output is normally in ANSI. If you want to redirect output in Unicode, you need to either use the /U startup option in *TCC*, or the Unicode Output option in TCMD.INI.

In the descriptions below, *filename* means either the name of a file or of an appropriate device (**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-RT (see below for additional redirection options using numeric file handles):

- ▶ Input redirection
- ▶ Output redirection
- ▶ Special considerations for specific commands
- ▶ NoClobber
- ▶ Multiple redirections
- ▶ Creating an empty file
- ▶ Redirection by handle
- ▶ "Here-document" redirection
- ▶ "Here-string" redirection

Input redirection

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

Output redirection

	overwrite	*append*
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 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). Redirection to the clipboard is always done using UTF16 Unicode.

If you redirect the output of a single internal command like DIR, the redirection ends automatically when that command is done. If you start a batch file with redirection, all of the batch file's output is redirected, and redirection ends when the batch file is done. Similarly, if you use redirection after the closing parenthesis of a command group (e.g., ...) **> report)**, all of the output from the command group is redirected, and redirection ends when the command group is done.

You can change the format of the redirected output. These options will override the UnicodeOutput and UTF8Output directives in TCMD.INI. Note: these options only work for redirecting output from TCC-RT internal commands and batch files.

`>:a`	Redirected output (STDOUT and/or STDERR) is ANSI (8 bit characters)
`>:u`	Redirected output is UTF16 Unicode
`>:8 or >:u8`	Redirected output is UTF8
`>>:a`	Appended redirected output (STDOUT and/or STDERR) is ANSI (8 bit characters)
`>>:u`	Appended redirected output is UTF16 Unicode
`>>:8 or >>:u8`	Appended redirected output is UTF8

Special considerations for specific commands

You cannot redirect all output from the execution of a DO loop due to the restriction that the DO command and its matching ENDDO may not be part of a command group.

To redirect the output of a TEXT command, append the redirection syntax to the TEXT command.

When you execute a FOR or GLOBAL 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 as in the example below:

```
( for /r %f in (*.btm) echo %@full[%f] ) > c:\temp\btmlst
```

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 /N1 command, the **Protect redirected output files** setting on the Startup tab of the configuration dialogs, or the Protect redirected output file 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, use **>!*filename***.

Redirection by handle

In addition to the redirection options above, TCC-RT also supports the CMD 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.

"Here-string" redirection

The "here-string" lets you send string text directly to a program's input. The syntax is:

```
program <<< string
```

This is similar to using KEYSTACK, but easier to enter for text input. (If you need to send special keys or insert waits, you'll need to use KEYSTACK.) For example, to send a string to the standard input of program:

```
c:\test\program.exe <<< This is some input text.
```

4.7.1.2 Pipes

Piping is a special form of redirection, using an additional instance of TCC-RT 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	*command1 | command2*
merge standard output and standard error	*command1 |& command2*
standard error only	command1 |&| command2

For example, to take the output of the ALIAS 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
```

The TEE and Y commands are "pipe fittings" which add more flexibility to pipes.

TCC's output is normally in ANSI. If you want to redirect output in Unicode, you need to either use the /U startup option in *TCC*, or the Unicode Output option in TCMD.INI.

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-RT implements pipes by starting a new process for the receiving program. This process goes through the standard shell start-up procedure, including execution of the TCSTART 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 file may cause the new process to execute in a different directory. When you use more than one pipe in a single command, e.g. the second example above with LIST, each pipe adds another instance of TCC-RT. If you need to execute the pipe in the same context, use in-process pipes (see below).

In-Process Pipes

In-process pipes work like the old-style DOS pipes, by creating a temporary output file, redirecting STDOUT to that file, and then redirecting the temp file to STDIN of the following command. The syntax is:

```
command1 |! command2
```

This the same as doing:

```
command1 > temp.dat & command2 < temp.dat
```

but is easier to type & to read.

The advantage of in-process pipes is that *command2* will be run in the same context as *command1*, so you can do things like modify environment variables without having them discarded when *command2* exits. There are also some disadvantages to using this type of "pseudo-pipe" -- it will usually be slower than a true pipe; it will use some disk space for its temp file; and *command2* will not be started until *command1* has exited.

ANSI, Unicode, and UTF-8 Output

You can change the format of output sent to a pipe. These options will override the UnicodeOutput and UTF8Output directives in TCMD.INI. The piped output options also work with in-process pipes (i.e., |

!:u). Note: these options only work for redirecting output from TCC-RT internal commands and batch files.

```
|:a           Piped output is ANSI
|:u           Piped output is UTF16 Unicode
|:8 or I:u8   Piped output is UTF8
```

4.7.2 Keystack

The KEYSTACK 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 lets you do exactly that.

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

To place the letters, digits, and punctuation marks you would normally type for your program into the KEYSTACK 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:

```
keystack F1 Enter
```

See Keys and Key names for details on how key names are entered. See the KEYSTACK 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 for additional details; see ACTIVATE for information on activating a specific window.

4.7.3 Page and File Prompts

Page Prompts

Several TCC-RT 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-RT is displaying information in page mode, for example with a DIR /P or SET /P command, it displays the message

```
Press ESC to quit, A to turn off paging or another key to continue...
```

At this prompt, you can press **Esc**, **Ctrl-C**, or **Ctrl- Break** if you want to quit the command. Pressing **A** will turn off the pause and prompt at the end of each page, and continue with 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 /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 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.

4.8   Tutorials

Video Tutorials

We're working on a series of video **command prompt tutorials** for CMDebug and TCC-RT. They will be posted here as soon as they are available.

- Creating and Debugging Windows Batch Files

Basic Tutorials

Take Command is a rich environment that allows you complete control of your Windows systems. We have created some quick command prompt tutorials that describe the basic features of the Take Command environment and the TCC-RT scripting langue. You will come up learning curve faster if you take a few moments to look through these documents and videos.

- Scripting Language Basics
- Triggers (Event Monitoring)
- Take Command In the Internet World

4.8.1   Scripting Language Basics

The TCC-RT command processor is highly upwardly compatible with (and a replacement for) the default Windows command processor CMD.EXE. It is suitable for creating both simple and highly sophisticated batch programs. The language can also be used at the command prompt to create very powerful real-time manipulation of your computer.

Overview

TCC-RT has a huge set of capabilities. These capabilities are grouped into three categories:

• Internal Commands - These are the primary language constructs. Common commands include, DIR, COPY, MOVE, etc. TCC-RT gives you instant access to more than 230 internal commands. (Microsoft's CMD.EXE has fewer than 40 internal commands.).

Internal vs. External Commands

When we talk about an internal command, we mean the command is built into the Take Command program. With CMD, some commands, like XCOPY are actually separate programs. In PowerShell, the commands are generally external programs. PowerShell requires a separate .NET Framework to be installed on the computer for the commands to work.

• Internal Variable - Internal variables are special variables built into TCC-RT 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. Take Command provides more than 270 internal variables that can tell you a great deal about your computer and how it is operating. These include installed hardware, hardware status, operating system and software status, etc.

• 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 and batch files to check on available system resources, manipulate strings and numbers, and work with files and filenames. There are more than 360 variable functions built into *TCC-RT*.

We are not going to talk about all of the features of the *TCC-RT* Language in this tutorial. (The manual is 1,300 pages long!) We are going to assume you know the basics of CMD and point you at a few of things that *TCC-RT* does better with less work than CMD.

<u>**Internal Commands**</u>

There are several aspects of *TCC-RT*'s internal command set that are definitely worth looking at:

- • Switches
- • Flow of Control Commands
- • KEYSTACK Command
- • HTTP and FTP
- • Event Monitoring Commands (Triggers) -- We made this into a separate tutorial

Each of these is covered below:

1. **Switches**

Switches modify commands by giving them special instructions. *TCC-RT* has a superset of the CMD switches and is generally compatible. We say generally, because unfortunately, CMD has not been consistent from version to version.
For example, in CMD, the COPY command has 7 switches (XCOPY has more). The *TCC-RT* COPY command has 34 switches. Examples of a few of the switches that the CMD COPY command does not have include:

- • /N Executes the copy command and shows you what the output would be, but does not actually execute the command
- • /O Copy the source file only if the target does not exist
- • /S Copy the subdirectory tree starting with the files in the source directory plus each subdirectory
- • /H Copy all matching files including those with a hidden or system attribute set
- • /W Delete files in the target directory that don't exist in the source directory

These switches allow you to custom tailor the language in ways that you cannot do with CMD. They perform very powerful operations with only two or three keystrokes.

2. **Flow of Control Commands**

One of the weakest areas of CMD is flow of control. These are the constructs like IF..THEN..ELSE or DO LOOPS that allow you to develop sophisticated batch programs. If you are creating data center batch processes, the limitations in CMD keep you from doing anything sophisticated.

TCC-RT provides a very rich set of constructs that allow you to duplicate (or exceed!) the capabilities of the typical Linux shells.

The following examples shows some of the types of DO Loops you can create:

Do Loops

DO count
DO FOREVER
DO varname = start TO end [BY step]
DO WHILE condition
DO UNTIL condition
DO UNTIL DATETIME date time
DO FOR n [SECONDS | MINUTES | HOURS]
DO varname IN [range...] [/I:"text" /S[n] /A:[-|+]hsad] fileset
DO varname IN [/T"delimiters"] /L stringset
DO varname IN /C stringset
DO varname in /P command
DO varname IN @file

TCC-RT also provides a very powerful IF..THEN..ELSE construct through the IFF command.

If...Then...Else Constructs

IFF condition1 THEN
commandset1
[ELSEIFF condition2 THEN commandset2]
...
[ELSE
commandset3]
ENDIFF

The alias in this IFF 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`

This example shows how a SWITCH construct works. 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:

Switch Constructs

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

3 .**KEYSTACK**

KEYSTACK takes a series of keystrokes and feeds them to a program or command as if they were typed at the keyboard. (It has no equivalent in CMD.) KEYSTACK is most often used for programs started from batch files. For 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 causes the following:

• Starts Word,
• The /w switch causes a delay of 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 and KEYSTACK are issued on a single command line. This ensures that the keystrokes are sent to Word's window, not back to Take Command.

4. **FTP and HTTP**

TCC-RT's FTP and HTTP commands allow you to treat http and ftp sites as if they were local disk drives. This is a huge advantage over CMD. In Working in the Internet World, we show you how to use these commands to create practical remote monitoring applications.

In simplest form, you can act as if an FTP or HTTP site is a local disk. For example, to get a directory of the JP Software FTP site, you could use this command:

Dir ftp://ftp.jpsoft.com/*

The following example shows how to include an ftp user name and password:

Dir ftp://username:password@ftp.abc.com/mydir/*

You can reference internet sites for DIR, COPY, MOVE, DEL and other commands. These commands also work with secure versions of FTP and HTTP.

5. **Event Monitoring Commands (Triggers)**

One of the most powerful features in *TCC-RT* are the event monitoring commands. They allow you to watch a wide variety of activities on your computer and "trigger" processes into action to deal with or report on issues.

This is described fully in Using Triggers in Take Command. It's well worth the read.

Internal Variables

Internal variables are special variables built into *TCC-RT* 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.

There are more than 280 of them (CMD has less than 10). Key types of variables include:

- Hardware status
- Operating system and software status
- Dates and times
- Drives and directories
- Error codes
- Screen, color, and cursor
- Take Command status
- Compatibility

Here is a simple example of how to use a common variable called _DOW (Day Of Week):

if "%_DOW" == "Mon" call c:\cleanup\weekly.bat

This example calls another batch file if today is Monday.

Before we go on...

A Quick Note:

One of the great mysteries of the command line is the % sign. What does it do? When you see a % sign in front of a variable or function, it means that the parser should evaluate the function and replace the variable or function with its text value. So, in the last example, %_DOW is replaced with the result, which in this case is MON, TUE or whatever.

How about something more real-time that you can run in the background:

DO FOREVER
if "%_BATTERYPERCENT" LT 25" MSGBOX Battery is low
ENDDO

This command will loop forever checking the battery status and popup a message box if the battery charge is getting low. MSGBOX is actually a very powerful command in TCC-RT. Check it out in the help file.
Here is an example that checks to see if 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

Take a look at the list of internal variables by category in the help file.

Variable Functions

Variable functions are one of the most powerful features of *TCC-RT*. 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 and batch files to check on available system resources, manipulate strings and numbers, and work with files and filenames.

There are more than 380 Variable Functions grouped into 13 categories. They allow you to gather and manipulate system information in very powerful ways. (CMD has no variable functions.). Remember...they are all built-in.

- Binary buffers
- Dates and times
- Drives and devices
- File content
- File names
- File properties
- Input dialog boxes
- Monitoring
- Network properties
- Numbers and arithmetic
- Strings and characters
- System status
- Utility

Using functions, *TCC-RT* can read and write text files, as well as some specialty files, such as the Windows Registry or .ini files. In the example below, we are going to read a .csv file called names.csv (which is a text file with fields separated by commas). The file looks as follows:

Joe,100,joe@company.com
Jane,200,jane@company.com
Peter,400,peter@company.com

Our example will read this file a line at a time, and select the email address in each line. It will then echo them to the console.

```
set filename=%@expand[names*.csv]
do record in @%filename
set email=%@field[",",3,%record] echo %email
enddo
```

This code does the following:

- The first line of the example creates a variable with the full file and pathname of the .csv file using the @expand function.
- The second line uses a special case of the DO command to:
- Open the filename we set in the first line with an @filename function
- Create a line counter that it sets to one
- Set up a new variable called "record"
- Read the first line of text up to the CR and returns it to "record"
- The @field function picks the third field in the line (which contains the email address) using a "," as the field delimiter and places it in a variable called "email". The delimiter could be anything you wanted.
- The ECHO command outputs the email address to the console

• ENDDO returns the loop to the do statement, which increments the line counter to the next line. If it's the end of the file, it terminates the loop.

This particular example seems sort of limited, but we use a variant of it to process our orders, construct registration keys and email them to our users with a remarkably small amount of code.

4.8.2   Event Monitoring in TCC-RT

The *TCC-RT* command interpreter provides a set of "trigger" commands that allow you to monitor activities on your computer and to trigger your computer to take an action based on changes occurring in the computer. This tutorial teaches you how to use them.

Overview

TCC-RT features a number of internal commands to allow you to do real-time monitoring of your system. These commands include:

- **FOLDERMONITOR** - Monitor folder and/or file creation, modification, and deletion
- **EVENTMONITOR** - Monitor event logs
- **NETMONITOR** - Monitor network connections and execute a command when a network is connected or disconnected
- **PROCESSMONITOR** - monitor processes and execute a command when a process is started or ended
- **SERVICEMONITOR** - monitor Windows services and execute a command when a service is started, paused, or stopped
- **USBMONITOR** - monitor USB connections and execute a command when a device is connected or disconnected
- **FIREWIREMONITOR** - monitor FireWire connections and execute a command when a device is connected or disconnected
- **CLIPMONITOR** - monitor the Windows Clipboard activity and execute a command when the clipboard is modified.
- DATEMONITOR - Monitor the current Windows system date and time and execute a command when the date and time matches.
- DEBUGMONITOR - Monitor writes to the OutputDebugString API.
- DISKMONITOR - Monitor free disk space.
- REGMONITOR - monitor Windows Registry keys
- SCREENMONITOR - Monitor the Windows screen saver.
- BLUETOOTHMONITOR - Monitor Bluetooth connections and execute a command when a device is connected or disconnected.
- POWERMONITOR - Monitor Windows system power changes.

Using these commands, you can easily watch most activity going on in your computer and provide alerts, such as emails or take actions, such as triggering a batch process if a monitored event occurs.

You can have up to 100 monitoring commands running simultaneously in a single Take Command tab window. The examples below show how simple it is to set up triggers and give you an idea about some of the things you can do with triggers.

Example 1 -- FOLDERMONITOR

FOLDERMONITOR lets you monitor directory and file creation, deletion, renaming, and modification. Let's say you want to watch for a file called "FinalResult.htm" to be created in the "d:\Results" subdirectory, and then copy it to "http://mycompany.com/results/FinalResult.htm"

The traditional approach would be to create a script file that waited forever for the file:

(*TCC-RT* Syntax) FINAL.CMD:

do forever
iff exist "d:\results\FinalResult.htm" then
copy "d:\results\FinalResult.htm" "http://mycompany.com/results/FinalResult.htm"
del FinalResult.htm
rem Wait for the file again
endiff
Delay 10
enddo

This creates a separate *TCC-RT* session, wasting memory and continuously requiring a small amount of CPU time.

In *TCC-RT* you can do the same thing with (on one line):

foldermonitor d:\results /i"FinalResult.htm" created forever
(copy "d:\results\FinalResult.htm" "http://mycompany.com/results/FinalResult.htm" &
del d:\results\FinalResult.htm)

Here is what is happening:

1. **Foldermonitor d:\results** -- causes the command to watch the subdirectory d:\results
2. **/i"FinalResult.htm"** -- says to include (watch) only files with the name FinalResult.htm in the monitoring
3. **created forever** -- means that we are looking only for files that are newly created and that we will do this in a continuous loop that will execute forever
4. **(copy "d:\results\FinalResult.htm" "http://mycompany.com/results/FinalResult.htm" & del d:\results \FinalResult.htm)** - will copy the new file to a website and deletes the file from the d:\results directory after it has been copied. You could execute a batch file here instead of creating a command group as we have done.

This command creates a separate thread in the current *TCC-RT* session.

FOLDERMONITOR also creates four environment variables when a file or folder is created, deleted, modified, or renamed that can be queried by the command. The variables are deleted after the command is executed.

- **_folderaction** -- The type of change to the file or folder. The possible values are:
  - CREATED
  - DELETED
  - MODIFIED This includes changing the file size, attributes or the date/time stamp.
  - RENAMED
- **_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

If you want to test for multiple changes, you should put the condition tests in a single FOLDERMONITOR command; otherwise FOLDERMONITOR will create a thread for each command (wasting your memory and CPU time).

For example, the following command will wait for any file to be created or changed in the d:\results directory and copy them to the web directory:

foldermonitor d:\results created modified forever (copy "%_folderfile1" "http://mycompany.com/results/")

Example 2 -- PROCESSMONITOR

PROCESSMONITOR monitors program starts and exits.

For example, if you want to be alerted with an email whenever a particular application exits:

processmonitor myapp* ended forever (sendmail bob@abc.com myapp Myapp just shut down!)

Here is what is happening:

1. **processmonitor myapp*** -- looks for any process with a name beginning with "myapp"
2. **ended forever** -- means that we are looking only for processes that have terminated (for any reason)
3. **(sendmail bob@abc.com myapp Myapp just shut down!)** - creates and sends an email using the internal TCC-RT Sendmail command to bob@abc.com with a subject of "myapp" and message text of ""myapp just shut down"

This is good for making sure that key production processes are operating as expected.

You can also use processmonitor to watch for specific processes being started. Maybe there is a virus that has escaped in your company that executes a malicious process -- call it malproc. The following script will look for the process running on a machine, kill it and send you an email identifying where the infection is.

processmonitor malproc started forever
(taskend /F malproc & sendmail bob@abc.com malproc I have malproc on my computer!)

This code does the following:

4. **processmonitor malproc** -- looks for any process with a name malproc
5. **started forever** -- means that we are looking only for processes that have just started (for any reason)
6. **(taskend /F malproc & sendmail bob@abc.com malproc I have malproc on my computer)** - uses the *TCC-RT* TASKEND command to force (/F) malproc to terminate immediately and then creates and sends an email using the internal TCC-RT Sendmail command to bob@abc.com with a subject of "malproc" and message text of ""I have malproc on my machine"

The *TCC-RT* triggers are exceptionally powerful and flexible commands that give you the ability to monitor and manage your computers like never before.

4.8.3   TCC-RT in the Internet World

The TCC-RT command interpreter has evolved to provide a variety of features that allow you to work in an Internet-centric world.

Overview

TCC-RT has the ability to:

- Access and Manipulate Remote Sites - You can get and put files in internet sites using several techniques including:
  - FTP (basic FTP)
  - TFTP (Trivial FTP)
  - FTPS (SSL FTP)
  - SFTP (SSH FTP)
  - HTTP (basic Web access)
  - HTTPS (SSL HTTP)
- Create Web Pages – *TCC-RT* allows you to construct web pages and populate them with real time data from your system

In this tutorial, we are going to show you a simple way to construct web pages with data from your computer and send them to a central website

Example 1 -- Create A Web Page

To create a web page, we use a very easy technique that was developed originally for Linux and has been implemented in *TCC-RT.*

The following script creates a web page (called status.html) and populates it with data about the status of your computer:

```
type <<- EndHTML >! status.html
<html>
<head>
<title>Server Status</title>
</head>
<body>
<h1>Server Status</h1>
<p>
Total memory: %@comma[%@winmemory[5]] bytes<br/>
Memory available: %@comma[%@winmemory[6]] bytes<br/>
Memory load: %@winmemory[0] %%</p>
<p>
Free disk space on drive C: %@diskfree[c:,Mc] MB<br/>
Free disk space on drive D: %@diskfree[d:,Mc] MB</p>
<p>
Reported generated %_isodate %_time by %@upper[%@filename[%_batchname]].</p>
</body>
</html>
EndHTML
```

Here is what is happening:

1. **<<-** -- This creates a redirect that sends everything in the following lines up to EndHTML to the type command -- which creates a text output

2. **>!** -- says redirect the text output of the type command to a file called status.html. The ! after the redirection command (>) means that the system should overwrite any existing status.html file

3. **HTML Code** -- The next few lines are standard HTML code that sets up some static header text

4. **System Data** -- The next few lines gather data from the system. The parser will examine each line of code and do variable expansion. What this means is that if you precede text with a % sign, *TCC-RT* will

assume that everything up to the next space is a variable or function and it will convert the variable or function to its actual value.

So, for example, %@winmemory[5] is evaluated as the actual amount of memory in the system.

%@winmemory[0] is evaluated as the amount of memory (as a percentage) actually being used.

You can nest functions, so %@comma[%@winmemory[5]] will apply the comma function to the amount of memory returned, properly formatting it.

The text below shows what is in the status.html file after running the program.

<html>
<head>
<title>Server Status</title>
</head>
<body>
<h1>Server Status</h1> <p>
**Total memory: 2,147,352,576 bytes
**
**Memory available: 2,064,941,056 bytes
**
Memory load: 61 %</p>
<p>
**Free disk space on drive C: 7,483 MB
 Free disk space on drive D: 207 MB</p> <p>**
Reported generated 2008-01-21 15:03:25 by BASICWEB.BTM.</p>
</body>
</html>

The following screenshot shows what the file looks like in a browser.
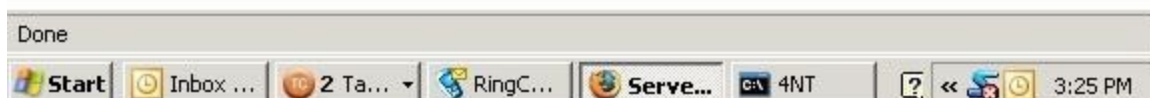
Example 2 -- Creating A Support System For Help Desks

If you have ever been part of a corporate help desk, you know that people often call in with issues about their computers, but you have no idea what is going on the computer. This example shows a simplified script you could put on all of the computers along with a desktop icon for users to press to execute the script. If a user has a problem this script will create a status web page and ftp: it to a central website so that the help desk group can get an idea what is going on in the computer.

We have expanded the code from the previous example:

```
type <<- EndHTML >! %_winname.html
<html>
<head>
<title>Server Status</title>
</head>
</body>
<h1>Server Status</h1>
<p>
Total memory: %@comma[%@winmemory[5]] bytes<br>
Memory available: %@comma[%@winmemory[6]] bytes<br>
Memory load: %@winmemory[0] %%</p>
```

```
<p>
Free disk space on drive C: %@diskfree[c:,Mc] MB<br>
Free disk space on drive D: %@diskfree[d:,Mc] MB</p>
<p>
Reported generated %_isodate %_time by %@upper[%@filename[%_batchname]].</p>
</body>
</html>
EndHTML
tasklist >> %_winname.html
services >> %_winname.html
copy %_winname.html ftp://user:[password@helpdesk.company.com/
```

In this example, we have added several features:

1. **%_winname** -- This is the name of the computer. It will create a unique file name

2. **Tasklist** -- The tasklist command outputs a list of all currently running processes. We are using >> to append the output of tasklist to the existing html file

3. **Services** - this is similar to the previous command, but in this case we are appending a list of system services to the html file

4. **Copy** -- the copy command is copying the html file to the company help desk website using ftp (with a user name and password used for security). Note that you can treat an ftp site as if it was a local directory - a great feature of *TCC-RT*.

This is a very simple example. *TCC-RT* includes hundreds of additional system variables and functions that can be used to gather status information.

In addition, if you understand Windows Management Interface (WMI), *TCC-RT* allows you to query anything known by WMI, which has almost all information about the status of the computer and activities going on in it.

5 Troubleshooting

- ▶ [Troubleshooting Service and Support](#)
- ▶ [Supported Platforms](#)
- ▶ [Error Messages](#)
- ▶ [Registration](#)

5.1 Troubleshooting, Service & Support

If you need help with *CMDebug*, we encourage you to review our documentation and then contact us for assistance if required.

If you need help with sales, ordering, or registration keys, please contact our Sales and Customer Service department. See [Contacting JP Software](#) 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 *CMDebug*, review the [Technical Support](#) 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.

5.1.1   Technical Support

Support Plans

Standard, no-charge support is available electronically through our Support Forums (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 https://jpsoft.com/.

Before you contact Technical Support, please review the What Information do we need? 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 Forums, where our support personnel can read and respond to your messages, and other users can participate in and benefit from the exchange. The Forums are 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 Forums which we make accessible as a mailing list 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 need 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 https://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 Forums 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).

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 *CMDebug* to run properly, review the information on Error Messages, and look through the Support Forum for any last-minute information.

If you need help with sales, ordering, 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 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 command to determine the *CMDebug* version and operating system version. This item is essential! Every question posted on the Forum should include a brief identification such as "*CMDebug* 25.0.10 under Windows 10 x64" 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.

- **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 *CMDebug* experiences an unrecoverable failure, it will save the error information to a *cmdebug.exception.log* file that contains the error info (including the file name, function, and line number of the error). The log files will be created in the installation directory if it is writeable (i.e. not in "Program Files" or "Program Files (x86)"). If not, they will be in "c:\programdata\JP Software".

5.1.2   Contacting JP Software

You can contact JP Software at the following addresses. 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

Online: Web site: https://jpsoft.com/
                                    Sales / Customer Service

Technical Support: Standard (no-charge) support: Available via our online Support Forum,
                                    accessible from the support area of our web site.

     See Technical Support 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.

5.2   Supported Platforms

CMDebug is a 32-bit or 64-bit GUI application.

TCC-RT is a 32-bit or 64-bit console (character-mode) application.

Both are designed to run under Windows 8, 2012, 10, 2016, and 2019.

The 32-bit and 64-bit versions of ***CMDebug*** are identical in features. The ***CMDebug*** installer will automatically choose the appropriate version for your version of Windows.

5.3   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.

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.

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 levels deep. Correct your alias list.

Already excluded files: You used more than one exclude range in a command. Combine the exclusions into a single range.

Array variable is already defined: You tried to create an array variable that already exists.

Bad disk unit: Generally caused by a disk drive hardware failure.

Batch file missing: TCC-RT 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-RT attempts to perform full path and filename expansion before copying to help ensure that files aren't inadvertently destroyed.

Can't create: TCC-RT 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-RT can't delete the specified file or directory. The disk is probably write protected.

Can't end current process: You attempted to terminate TCC-RT with a [TASKEND](#) command. TASKEND can only be used to end other processes; to terminate *TCC*, use the [EXIT](#) command.

Can't get directory: TCC-RT can't read the directory. The disk drive is probably not ready.

Can't make directory entry: TCC-RT 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-RT 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](#) 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: *TCC-RT* 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 (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 or DIRS can't find any entries in the directory stack.

Disk is write protected: The disk cannot be written to. Check the disk and remove the write-protect tab or close the write- protect window if necessary.

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 stand error more than once in the same command. Correct the command and try again.

Error in command line directive: You used the **//iniline** option to place an .INI directive on the startup command line, but the directive is in error. Usually a more specific error message follows, and can be looked up in this list.

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.

Exceeded the maximum number of simultaneous monitors: You have attempted to create more than 100 monitoring functions.

File Allocation Table bad: Windows can't access the FAT on the specified disk. This can be caused by a bad disk, a hardware error, or an unusual software interaction.

File association not found: The ASSOC command could not find a file association for the specified extension in the Windows registry.

File exists: The requested output file already exists, and TCC-RT won't overwrite it.

File not found: TCC-RT couldn't find the specified file. Check the spelling and path name.

File type not found: The FTYPE command could not find the specified file type in the Windows registry.

General failure: This is usually a hardware problem, particularly a disk drive failure or a device not properly connected to a serial or parallel port. Try to correct the problem, or reboot and try again. See also: **Data error** above.

Infinite COPY or MOVE loop: You tried to COPY or MOVE a directory to one of its own subdirectories and used the /S switch, so the command would run forever. Correct the command and try again.

Insufficient disk space: COPY or MOVE ran out of room on the destination drive. Remove some files and retry the operation.

Invalid array argument (out of bounds): You tried to reference an array element that exceeded the array size.

Invalid batch file: The batch file is corrupted, or improperly compressed, or encrypted. Retry with a new copy of the file.

Invalid count: The character repeat count for KEYSTACK is incorrect.

Invalid date: An invalid date was entered. Check the syntax and reenter.

Invalid drive: A bad or non-existent disk drive was specified.

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 time: An invalid time was entered. Check the syntax and reenter.

Label not found: A GOTO or GOSUB referred to a non-existent label. Check your batch file.

Listbox is full: There is no more room in the Find Files / Text dialog's results box. Use a more selective search, or use the FFIND command rather than the dialog.

Missing close paren: A KEYSTACK command is missing a closing parentheses around a character group. Correct the command.

Missing ENDTEXT: A TEXT command is missing a matching ENDTEXT. Check the batch file.

Missing GOSUB: TCC-RT cannot perform the RETURN command in a batch file. You tried to do a RETURN without a GOSUB, or your batch file has been corrupted.

Missing SETLOCAL: An ENDLOCAL was used without a matching SETLOCAL.

No aliases defined: You tried to display aliases but no aliases have been defined.

Not an array variable: You tried to reference a non-existent array variable.

No closing quote: *TCC-RT* couldn't find a second matching back quote [`] or double-quote ["] on the command line.

No expression: The expression passed to the %@EVAL variable function is empty. Correct the expression and retry the operation.

No shared memory found: The SHRALIAS command could not find any global alias list, history list, or directory history list to retain, because you executed the command from a session with local lists. Start TCC-RT with at least one global list, then invoke SHRALIAS.

No SMTP server: SENDMAIL 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 is not a directory.

Not an alias: The specified alias is not in the alias list.

Not in environment: The specified variable is not in the environment.

Not ready: The specified device can't be accessed.

Not same device: This error usually appears in RENAME. You cannot rename a file to a different disk drive.

Out of function space: You are attempting to create a User-defined Function 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 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 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 command to unload SHRALIAS.EXE, but it was never loaded. This message is informational and may not indicate an error condition.

String too long: You tried to put more than 2038 characters into the KEYSTACK buffer. Reduce the number of characters you are trying to send to the application at one time.

Syntax error: A command or variable function 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 variable function. Correct the expression and retry the operation.

UNKNOWN_CMD loop: The UNKNOWN_CMD alias 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-RT* 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).

Unknown option name: (OPTION) You are attempting to modify or display an invalid or unknown option name.

Unknown process: TASKEND cannot find the process you specified. If you are ending a process using the title you may need to use wildcards to get a match on the title string. Correct the command and try again.

Variable loop: A nested environment variable refers to itself, or variables are nested more than 16 deep. Correct the error and retry the command.

Window title not found: The ACTIVATE command could not find a window with the specified title. Correct the command or open the appropriate window and try again.

Write error: Windows encountered a disk write error; usually caused by a bad or unformatted disk. See also: **Data error** above.

6 Reference

▸ Colors, Color Names & Codes
▸ ASCII Codes and Key Names

6.1 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 **BRI**ght may be shortened to their first three letters. The available color names, shown below on approximations of the 8 basic background colors, are: **BLA**ck, **BLU**e, **GRE**en, **CYA**n, **RED, MAG**enta, **YEL**low, **WHI**te.

	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. **BRI**ght 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, 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 command also supports the CMD style color specification *bf*, where *b* and *f* are CMD.EXE's codes for background and foreground colors, respectively (shown in the CMD columns of the table below). The numeric values of these codes are the same as the *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 codes* (hexadecimal)		ANSI X3.64 codes (decimal)	
normal	bright		normal	bright	normal	bright	foreground	background
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 and native color codes are identical, the difference is in representation only.

Use one number to substitute for the **[BRIght] fg** portion of the color name, and a second to substitute for the **[BRIght] bg** portion. For example, instead of **bright white on red** you could use **15 on 4** to save space in a ColorDir specification.

The @OPTION 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, many 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 and they cannot be corrected using the *Take Command* color specifications. You can, however, define a custom color palette to get the exact colors you want, via the "Tab Colors" button on the Configure Take Command / Tab dialog.

6.2   ASCII Codes and Key Names

For ASCII codes and key names see:

- ASCII Tables
- Keys & Key Names

The remainder of this section gives a explanation of the ASCII character sets and key names. 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.

6.2.1   ASCII Tables

These tables show the 128-character ASCII set for U.S. English systems. Most of the characters in code range 32..126 (the only codes for which ASCII specifies displayable symbols) will be the same on non-U.S. systems. The symbols associated with all other codes vary from font to font, as well as from country to country.

For more details on ASCII, character sets, and key codes, see the general information topic on ASCII, Key Codes, and ANSI X3.64 Commands.

- Control Characters 0 - 31, 127
- Printing Characters 32 - 47
- Printing Characters 48 - 63
- Printing Characters 64 - 79
- Printing Characters 80 - 95
- Printing Characters 96 - 111
- Printing Characters 112 - 126

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
22	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

6.2.2   Key Names

Key names are used in *TCC-RT* in the INKEY and KEYSTACK commands. 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-F24, Tab, Bksp, Enter, Up, Down, Left, Right, PgUp, PgDn, Home, End, Ins, Del, Esc, Apps, Sleep, Select, Execute, Print, Mute, VolumeUp, VolumeDown
Alt-	A-Z, 0-9, F1-F24, Bksp, and the non-alphanumeric keys `-=[]\;',./
Ctrl-	A-Z, F1-F24, Tab, Bksp, Enter, Up, Down, Left, Right, PgUp, PgDn, Home, End, Ins, Del
Shift-	A-Z, F1-F24, Tab
LWin-	A-Z, F1-F24
RWin-	A-Z, F1-F24

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 *CMDebug* or *TCC-RT*. 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 read in INKEY or INPUT because *TCC-RT* never receives these keystrokes. However, KEYSTACK 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.

6.3   File Systems & File Names

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 when 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:

▶   Executable Files & File Searches

6.3.1 Executable Files & File Searches

When TCC-RT 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 an *.EXE* extension.)

If TCC-RT cannot find an executable program to run, it next looks for a matching batch file name. TCC-RT looks first for a *.BTM* file, then for a *.CMD* file, then for a *.BAT* file, and finally for a *.REX*, *.REXX*, *.PL*, *.PY*, *.RB, or .TCL* file (if REXX, Perl, Python, Ruby, and/or Tcl are enabled).

You can change the list of extensions that are considered "executable", and the order in which they are searched, with the PATHEXT environment variable, and the related PathExt configuration option. PATHEXT is supported for compatibility reasons but should not generally be used as a substitute for executable extensions, which are more flexible.

Note: If the search for an external program or batch file fails, TCC-RT checks to see if the command name matches the name of a file with an executable extension. If an executable extension is found, TCC-RT runs the program specified when the association was defined. If no executable extension is found, TCC-RT 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-RT* calls the ShellExec Windows API. TCC-RT 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-RT 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-RT (and some applications) search for executable files. For example, if you wanted TCC-RT 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 command. You can use the ESET 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-RT 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-RT then checks for a corresponding **App Paths** entry in the Windows registry (either in the HKCU or HKLM tree). **App Paths** entries are created by some applications during the installation process.

Remember, TCC-RT 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 command for details.)

If you include an extension as part of the command name, TCC-RT only searches for a file with that extension. Similarly, if you include a path as part of the command name, TCC-RT 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-RT to recognize this approach, you can set the Unix/Linux Paths configuration option.

Once the file is found, TCC-RT executes it based on its extension. *.EXE* 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-RT (*.EXE, .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] image1.psp
```

and *Take Command* will request that Windows start the application for you. See Windows File Associations for additional details on how to control Windows file associations in TCC-RT.

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-RT Search Sequence
WP	Search for any executable file whose base name is *WP*.
WP.EXE	Search for *WP.EXE*; will not find files with other extensions.
C:\WP\WP	Looks in the *C:\WP* directory for any executable file whose base name is *WP*. Does not check the standard search directories.
C:\WP\WP.EXE	Looks only for the file *C:\WP\WP.EXE*.
LAB.DOC	Search for *LAB.DOC*, if *.DOC* is defined as an executable extension. Runs the associated application if the file is found. If *.DOC* is not an executable extension, passes the name to Windows to check for a Windows file association.
C:\LI\LAB.DOC	Looks only for the file *C:\LI\LAB.DOC*, and only if *.DOC* is defined as an executable extension. Runs the associated application if the file is found. If

.DOC is not an executable extension, passes the name to Windows to check for a Windows file association.

If the first argument on a command line is in the format "env_var=value command options" (and env_var=value doesn't match an external command) then TCC-RT will set the specified environment variable to the value, execute the command, and then remove the variable.

If TCC-RT 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 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). If *TCC-RT* cannot find an **UNKNOWN_CMD** alias, it will look for a plugin command named **UNKNOWN_CMD***.* Otherwise, TCC-RT displays an "Unknown command" error message and waits for your next instruction.

See also: the WHICH command.

6.3.2 Windows File Associations

Windows includes the ability to associate file extensions with specific applications. 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-RT* first searches for an external program file with a standard extension (*.EXE*, *.CMD,* etc.). It then checks executable extensions. If all of these tests fail, TCC-RT passes the command name to Windows to see if Windows can find an association for it.

TCC-RT 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 command modifies or displays the associations between extensions and file types in the Windows registry. The FTYPE command modifies or displays the default command used to "open" a file of a specified type.

Executable extensions defined in *TCC-RT* always take precedence over file associations defined in Windows. For example, if you associate the *.TXT* extension with your own editor using a *TCC-RT* 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-RT*.

See also: START, ASSOC, FTYPE, Executable Extensions, Executable Files and File Searches.

6.3.3 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 (now largely obsolete).

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 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 for details. Additional information about disk files and directories is available under Directories and Subdirectories, File Names, and File Attributes.

6.3.4   File Systems

CMDebug and *TCC-RT* use only documented Windows APIs to access the file systems, so they work with any file system supported by Windows.

Additional information about disk files and directories is available under Drives and Volumes, Directories and Subdirectories, File Names, and File Attributes.

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. *TCC-RT* 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 *TCC-RT*.

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. TCC-RT 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, *TCC-RT* 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.

TCC-RT also allows you to use UNC directory names when changing directories.

OpenAFS

TCC-RT 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.

6.3.5   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).

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-RT*, the path and filename can be up to 32,767 characters, though many Windows applications (including CMD and Explorer) have trouble with path and filename lengths exceeding 260 characters. Shorter paths and names are advisable under Windows whenever feasible.

TCC-RT 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 [**.**] means "the current default directory." Two periods [**..**] means "the directory which contains the current default directory" (referred to as the parent directory). These special names can be used wherever a full directory name can be used. *TCC-RT* allows you to use additional periods to specify directories further "up" the tree (see Extended Parent Directory Names).

Additional information about disk files and file systems is available under Drives and Volumes, File Systems, File Names, and File Attributes.

6.3.6 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](#), 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-RT*.

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](#) for details on avoiding problems with interpretation of those file names under *TCC-RT*.

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 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-RT*, 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, File Systems, Directories and Subdirectories, File Attributes, and Time Stamps.

6.3.7   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, Junction / Symbolic Link / Reparse point, No Scrub, and Integrity. *TCC-RT* fully supports these extended attributes.

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-RT* 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 *TCC-RT*, 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). *TCC-RT* does not directly modify the volume label or any of its attributes, and provide read access only through the VOL command and the @LABEL[] 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 available only on an NTFS volumes) are reset. It is not stored by the file system. When *TCC-RT* 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 and DIR commands, and by the @ATTRIB and @WATTRIB variable functions.

Attributes can be set, reset, and viewed with the ATTRIB command. The DIR command also has options to view the attribute status of files, and to view information about normally invisible hidden and system files and directories.

6.3.8 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-RT** 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 seconds. NTFS volumes have a 100 nanosecond resolution for the file creation and last write. (UNIX and Linux systems use 1 second 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-RT** 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.

file system	resolution	earliest time stamp	latest time stamp
FAT/VFAT	2 s	1980-01-01 00:00:00 *local*	2107-12-31 23:59:58 *local*
NTFS	100 ns	1601-01-01 00:00:00 *UTC*	60056-05-28 *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 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, File Systems, Directories and Subdirectories, and File Names.

6.3.9   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.

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

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

6.4   Regular Expression Syntax

Onigmo Regular Expressions Version 6.2.0

This section covers the Ruby regular expression syntax. For information on Perl regular expression syntax, see your Perl documentation or https://perldoc.perl.org/perlre.html.

⊟ **Syntax elements**

```
\         escape (enable or disable meta character meaning)
|         alternation
(...)     group
[...]     character class
```

⊟ **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)

- **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,

+ works on UTF8, UTF16, UTF32

\R   Linebreak

Unicode:
  (?>\x0D\x0A|[\x0A-\x0D\x{85}\x{2028}\x{2029}])

Not Unicode:
  (?>\x0D\x0A|[\x0A-\x0D])

\X   eXtended grapheme cluster

Unicode:
  (?>\P{M}\p{M}*)

Not Unicode:
  (?m:.)

⊟ 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*)/

⊟ 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 (*)

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 ([:xxxxx:], negate [:^xxxxx:])

Not Unicode Case:

alnum alphabet or digit char
alpha alphabet
ascii code value: [0 - 127]
blank \t, \x20
cntrl
digit 0-9
graph include all of multibyte encoded characters
lower
print include all of multibyte encoded characters
punct
space \t, \n, \v, \f, \r, \x20
upper
word alphanumeric, "_" and multibyte characters
xdigit 0-9, a-f, A-F

Unicode Case:

alnum Letter | Mark | Decimal_Number
alpha Letter | Mark
ascii 0000 - 007F
blank Space_Separator | 0009
cntrl Control | Format | Unassigned | Private_Use | Surrogate
digit Decimal_Number
graph [[:^space:]] && ^Control && ^Unassigned && ^Surrogate
lower Lowercase_Letter
print [[:graph:]] | [[:space:]]
punct Connector_Punctuation | Dash_Punctuation | Close_Punctuation | Final_Punctuation | Initial_Punctuation | Other_Punctuation | Open_Punctuation
space Space_Separator | Line_Separator | Paragraph_Separator | 0009 | 000A | 000B | 000C | 000D | 0085
upper Uppercase_Letter
word Letter | Mark | Decimal_Number | Connector_Punctuation
xdigit 0030 - 0039 | 0041 - 0046 | 0061 - 0066 (0-9, a-f, A-F)

Extended groups

(?#...) comment

(?imxdau-imx)   option on/off
                i: ignore case
                m: multi-line (dot(.) match newline)
                x: extended form

              character set option (character range option)
                d: Default (compatible with Ruby 1.9.3)
                    \w, \d and \s doesn't match non-ASCII characters.
                    \b, \B and POSIX brackets use the each encoding's rules.
                a: ASCII
                    ONIG_OPTION_ASCII_RANGE option is turned on.
                    \w, \d, \s and POSIX brackets doesn't match non-ASCII characters.
                    \b and \B use the ASCII rules.
                u: Unicode
                    ONIG_OPTION_ASCII_RANGE option is turned off.
                    \w (\W), \d (\D), \s (\S), \b (\B) and POSIX brackets use the each
                    encoding's rules.

(?imxdau-imx:subexp) 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.

\K   keep
              Another expression of look-behind. Keep the stuff left of the \K, don't
              include it in the result.

(?>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.

(?(cond)yes-subexp), (?(cond)yes-subexp|no-subexp)
            conditional expression
            Matches yes-subexp if (cond) yields a true value, matches no-subexp otherwise.

Following (cond) can be used:

(n) (n >= 1)
   Checks if the numbered capturing group has matched something.

(<name>), ('name')
   Checks if a group with the given name has matched something.

⊟ Back reference

\n	back reference by group number (n >= 1)
\k<n>	back reference by group number (n >= 1)
\k'n'	back reference by group number (n >= 1)
\k<-n>	back reference by relative group number (n >= 1)
\k'-n'	back reference by relative 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 setted.

Back reference with nest level

   level: 0, 1, 2, ...

   \k<n+level>   (n >= 1)
   \k<n-level>   (n >= 1)
   \k'n+level'   (n >= 1)
   \k'n-level'   (n >= 1)
   \k<-n+level>   (n >= 1)
   \k<-n-level>   (n >= 1)
   \k'-n+level'   (n >= 1)
   \k'-n-level'   (n >= 1)

   \k<name+level>
   \k<name-level>
   \k'name+level'
   \k'name-level'

Destinate relative nest level from back reference position.

example 1.

   /\A(?<a>|.|(?:(?<b>.)\g<a>\k<b+0>))\z/.match("reer")

example 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

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)
\g<0>     call the whole pattern recursively
\g'0'     call the whole pattern recursively
\g<-n>    call by relative group number (n >= 1)
\g'-n'    call by relative group number (n >= 1)
\g<+n>     call by relative group number (n >= 1)
\g'+n'     call by relative 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"

Perl syntax:: use (?&name), (?n), (?-n), (?+n), (?R) or (?0) instead.

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

Syntax dependent options

+ RUBY
(?m): dot(.) match newline

+ PERL, JAVA, and Python
(?s): dot(.) match newline
(?m): ^ match after newline, $ match before newline

+ PERL
(?d), (?l): same as (?u)

Original extensions

+ hexadecimal digit char type \h, \H
+ named group   (?<name>...)
+ named backref   \k<name>
+ subexp call   \g<name>, \g<group-num>

Missing features compared with Perl 5.14.0

+ \N{name}, \N{U+xxxx}, \N
+ \l,\u,\L,\U, \C
+ \v, \V, \h, \H, \o{xxx}
+ (?{code})
+ (??{code})
+ (?|...)
+ (*VERB:ARG)

* \Q...\E
This is effective in PERL and JAVA.

Disabled functions by default syntax

+ capture history

(?@...) and (?@<name>...)

ex. /(?@a)*/.match("aaa") ==> [<0-1>, <1-2>, <2-3>]

Problems

+ Invalid encoding byte sequence is not checked.

ex. UTF-8

* Invalid first byte is treated as a character.
/./u =~ "\xa3"

* Incomplete byte sequence is not checked.

/\w+/ =~ "a\xf3\x8ec"

6.5 Plugins

TCC plugins are user-written DLL's that allow you to write your own internal variables, variable functions, and internal commands, 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](#)) are executed.

You can specify a particular plugin to execute by prefixing the function / variable / command name with the plugin name and a $. For example:

```
echo %_myplugin$variable
echo %@myplugin$func[abc]
myplugin$mycommand
```

Plugins can be written in any language that can create a Windows DLL. The **TCC** plugin SDK has samples for Visual C++ and Delphi. The SDK is available on our web site at https://jpsoft.com/downloads/sdk/sdk.zip.

Keystroke Plugins:

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. Keystroke plugins are called after key aliases, and before **TCC** looks for the default action for that key.

V24+ only: If the value passed in "nKey" is 0, the key is not a valid Unicode character, and the plugin needs to parse the **pszKey** string to get the name. The name will be passed in the format:

```
[Ctrl-][Alt-][Shift-]key
```

For example:

```
F12
Ctrl-F1
Ctrl-Alt-Left
Ctrl-Shift-F5
```

The keystroke plugin can modify the **nKey** or **pszKey** value and pass it back to **TCC** to evaluate the default action for the (new) value. If **nKey** is != 0, **TCC** will treat it as a normal Unicode character. If **nKey** = 0, **TCC** will evaluate **pszKey** for a valid keyname.

If the plugin handled the key and doesn't want **TCC** to do anything more, set **nKey** to 0 and **pszKey** to an empty string (write a null to the first byte).

Plugin Syntax:

```
// PluginInfo structure – returned by plugin in response to GetPluginInfo()
call from command processor
```

```
// Note that the strings should all be Unicode; if your PlugIn is compiled
for ASCII you'll need to use
//   the MultiByteToWideChar API to convert the strings before passing them
back to TCC
typedef struct {
      TCHAR      *pszDll;         // name of the DLL
      TCHAR      *pszAuthor;      // author's name
      TCHAR      *pszEmail;       // author's email
      TCHAR      *pszWWW;         // author's web page
      TCHAR      *pszDescription; // (brief) description of plugin
      TCHAR      *pszFunctions;   // comma-delimited list of functions in
the
                                  //   plugin (leading _ for internal
vars, @ for
                                  //   var funcs, * for keystroke
function,
                                  //   otherwise it's a command)
      int        nMajor;         // plugin's major version #
      int        nMinor;         // plugin's minor version #
      int        nBuild;         // plugin's build #
      HMODULE    hModule;        // module handle
      TCHAR      *pszModule;     // module name
} PLUGININFO, *LPPLUGININFO;


// structure passed to plugin functions to monitor keystrokes. A
//   keystroke function can be named anything, but must prefix a
//   * to its name in the function list (pszFunctions, above).
//   If the keystroke plugin handled the keystroke and doesn't want
//   to pass it back to TCC, it should set nKey = 0 and pszKey to an empty
string.
//   The command processor will call the keystroke function with all
//     parameters set to 0 just before accepting input for each new
//     command line.
//   The string pointers are Unicode
typedef struct {
      int        nKey;           // key entered
      int        nHomeRow;       // start row
      int        nHomeColumn;    // start column
      int        nRow;           // current row in window
      int        nColumn;        // current column in window
      LPTSTR     pszLine;        // command line
      LPTSTR     pszCurrent;     // pointer to position in line
      int        fRedraw;        // if != 0, redraw the line
      LPTSTR     pszKey;         // (v24+ only) ASCII name of key (for
example, "Ctrl-Alt-Home")
} KEYINFO, *LPKEYINFO;
```

```
__declspec(dllexport) BOOL WINAPI InitializePlugin( void );      // called
by command processor after loading all plugins
__declspec(dllexport) LPPLUGININFO WINAPI GetPluginInfo( HMODULE hModule );
                                                                // called
by command processor to get information from plugin, primarily for the
names of functions & commands
__declspec(dllexport) BOOL WINAPI ShutdownPlugin( BOOL bEndProcess );  //
called by command processor when shutting down
                                                                //   if
bEndProcess = 0, only the plugin is being closed
                                                                //   if
bEndProcess = 1, the command processor is shutting down
```

The functions listed in "pszFunctions" and called by TCC need to be in the format:

DLLExports INT WINAPI MyFunctionName(LPTSTR pszArguments);

Internal variable names in pszFunctions (and their corresponding functions) must begin with an underscore ('_').

Variable function names in pszFunctions must begin with an @; the corresponding function must be prefixed by "f_". (This allows variable functions to have the same name as internal commands.)

For example:

pszFunctions = "reverse,@reverse"

Entering the name "reverse" on the command line will invoke the command reverse()
Entering the name "@reverse[]" on the command line will invoke the variable function f_reverse()

Variable function names are limited to a maximum of 31 characters.

Internal command names are any combination of alphanumeric characters (maximum 12 characters).

Calling the PlugIn:

For internal variables, pszArguments is empty (for output only)

For variable functions, pszArguments passes the argument(s) to the plugin function

For internal commands, pszArguments is the command line minus the name of the internal command

Returning from the PlugIn:

For internal variables and variable functions, copy the result string over pszArguments. The maximum string length for internal variables and variable functions is 32K (32767 characters + terminating null character).

Internal variables have no meaningful integer return value. For variable functions, the integer return can be:

0 = success

< 0 = failure; error message already displayed by the PlugIn function

> 0 = failure; error value should be interpreted as a system error and
displayed by 4NT / TC

There is a special return value (0xFEDCBA98) that tells the parser to assume that the plugin decided not to handle the variable/function/command. The parser then continues looking for a matching internal, then external. Note that you can use this return value to have your plugin modify the command line and then pass it on to an existing internal variable/function/command!

For internal commands, return the integer result (anything left in pszArgument will be ignored)

Exception Handling:

TCC will trap any exceptions occurring in the plugin, to prevent the plugin from crashing the command processor. An error message will be displayed and the plugin will return an exit code = 2.

Filename Completion

When TCC is performing filename ("tab") completion, it will look for a plugin function named TABCOMPLETION. Like TABCOMPLETE scripts, TABCOMPLETION allows you to create plugin functions to customize TCC's filename completion. The syntax is:

```
INT WINAPI TABCOMPLETION(LPCTSTR Command, LPCTSTR Argument, int
Index, LPCTSTR CommandLine);
```

Command - the name of the command at the beginning of the command line

Argument - the current argument being evaluated

Index - the offset in the command line of the beginning of Argument

CommandLine - the entire command line (double quoted)

When the plugin function finishes, it should return 0 if it processed the completion, and save the result(s) in the TABCOMPLETIONRESULT environment variable. If the function has multiple completion results, they should be added to TABCOMPLETIONRESULT, separated by a space (and double quoted if they contain any whitespace).

TCC will examine the contents of TABCOMPLETIONRESULT; if it contains a single value TCC will insert it at the completion point on the command line. If there are multiple return values, TCC will display a popup window for selection (like the F7 completion window).

TCC will try to find a filename completion script first; if none of them perform the requested completion, TCC will look for the plugin function.

You can specify a particular plugin to execute by prefixing the function / variable / command name with the plugin name and a $. For example:

```
echo %_myplugin$variable
echo %@myplugin$func[abc]
myplugin$mycommand
```

6.6   Limits

Most *TCC-RT* arguments are only limited by the amount of available RAM. There are a few (like the maximum filename size) that are limited by the Windows APIs.

Length Limits (characters)

entity	name	value	combined
environment variable	none	none	none
alias	none	none	none
user defined variable function	none	none	none

command type	before expansion	after expansion
command line	none	none
command group	none	none

Nesting Limits

command	depth
CALL	no limit
DO	no limit
FOR	no limit
GOSUB without parameters	no limit
GOSUB with parameters	22
SETLOCAL	32
IFF	no limit

Miscellaneous Limits (characters)

entity	limit
character count in any function	none
number of batch file parameters	8,191
number of GOSUB parameters	255
file name (Windows limitation)	32,767
include list	none
single parameter	none
global alias list *	262,144
global function list *	131,072
directory stack (PUSHD)	16,383

* The global alias list and global function list sizes may be increased with the AliasSize and FunctionSize .INI directives.

7   Copyright & Version

jpsoftware

CMDebug and ***TCC-RT*** 32 for Microsoft Windows 10 / Windows 11 / Server 2016 / Server 2019 / Server 2022

Software: Copyright © 2024, Rex Conn and JP Software Inc.
All Rights Reserved.

Version 32 Help System
Help text: Copyright © 2024 JP Software Inc.
All Rights Reserved.

Language translations by Christian Albaret (French), Hans-Peter Grözinger and Klaus Meinhard (German), Stefano Piccardi and Rodolfo Giovanninetti (Italian), Dmitry Yerokhin (Russian), and Orlando Hevia (Spanish).

TPIPE is a licensed version of the TextPipe Engine http://datamystic.com

The Scintilla edit control is Copyright 1998-2023 by Neil Hodgson http://www.scintilla.org

We gratefully acknowledge the contributions of Charles Dye, Vincent Fatica, and our other users.

This help material was last revised on Thursday, April 11, 2024

Index

- D -

- E -

- F -

- M -

- N -