4NT 8.0 and Take Command 8.0

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

Table of Contents

ACKNOWLEDGEMENTS

We couldn't produce products like 4NT and Take Command without the dedication and quality work of many people. We can't list all of our beta testers here! A special thanks to all of you who helped make 4NT and Take Command elegant, reliable, and friendly.

1 Overview

4NT 8.00
TAKE COMMAND 8.00

Welcome to our help! We have designed this material to accompany our current 588 Windows products: ***4NT*** and ***Take Command*** (usually abbreviated ***TC***). ***4NT*** and ***Take Command*** are designed for Windows 2000, Windows XP, Windows 2003, and Windows Vista.

Each of these programs is a **command interpreter** or **shell**. That means that they respond to commands you type at the prompt. They are also able to process batch files 162 containing sequences of commands.

4NT is designed as a powerful alternative to CMD.EXE**,** the default command processor in Windows 2000 / XP / 2003 / Vista.

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

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

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

2 Basics

This section provides a general description of ***4NT*** and ***Take Command*** operation.

2.1 Starting the Command Processor

You will typically start the command processor from a Windows shortcut, located:

- on the desktop, or
- in the Quick Launch bar, or
- 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 command processor folder or group (in the **Programs** section of the **Start** menu) and a desktop object (shortcut) which starts the command processor. Usually these are sufficient, but if you prefer, you can create multiple desktop objects or items to start the command processor with different startup commands or options, or to run different batch files or other commands. See Desktop Integration 10 for details. You can use these items to run commonly-used commands and batch files directly from the desktop.

Each item or icon represents a different command processor window. You can set any necessary command line parameters for the command processor such as a command to be executed, any desired switches, and the name and path for the .INI file 91. See Command Line Options 4 for more information on startup command line switches and options.

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

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

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

When the command processor starts, it automatically runs the optional TCSTART 8 or 4START 8 program, as described in Automatic Startup and Termination Programs 8. You can use that to load

aliases, functions, and environment variables, and to otherwise initialize the command processor environment.

You can also place the name of a batch file, internal or external command, or alias at the end of the Command Line field for any item (as shown in the example above). The batch file, command, or alias will be executed after TCSTART 8 or 4START 8, but before the first prompt is displayed.

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

The command processor startup command line does not need to contain any information. When invoked with an empty command line, the command processor will configure itself from the .INI file 91, run TCSTART 8 or 4START 8, and then display a prompt and wait for you to type a command. However, you may add information to the startup command line 4 that will affect the way the command processor operates.

2.1.1 Startup Command

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

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

```
c:\4NT\4NT.exe @c:\4NT\4NT.ini
```

Although the startup command line is usually very simple, you can add several options. You can do this manually in the Windows **RUN** dialog, in a Windows shortcut file (*.LNK)*, at the command processor prompt or in a batch file (with or without using the internal START 364 command). Each of these methods will start a new instance of the selected command processor, which will run in a new window, except when ***4NT*** is started from ***4NT*** (either at the command prompt or within a batch file) without the START 364 command.

When you use a pipe 39 in a command, either at the command prompt or in a batch file, the command processor starts another instance of itself, using the same command line parameters (except as required for the pipe).

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

```
d:\path\PROGRAM.EXE [d:\path] [[/]@d:\path\inifile] [//directive=value...] [/H
   /L: /LA /LD /LF /LH /Q /S /T:bf /U /V /X ] [/C | /K] [command]
```

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

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

The command line must start with the path and name of the executable program file (**TCMD.EXE** or **4NT.EXE**):

d:*path**PROGRAM*.EXE

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

d:*path*

If included, this second copy *d:\path* of the command processor's 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. ***4NT*** and ***TC*** can find their own directory without a **COMSPEC** path, and usually the **COMSPEC** variable should be left pointing to the default character mode command processor in use on your system.

@*d*:*path**inifile* ***OR***
/@*d*:*path**inifile*

This option sets the path and name of the .INI file 91. You don't need this option if

1) your .INI file 91 is named *TCMD32.INI* ***(TC)*** or *4NT.INI* ***(4NT),*** and
2) it is in one of the following directories:
   2.1) the same directory as the command processor
   2.2) for ***TC:*** the ***WINDOWS*** directory (***%SystemRoot***)
   for ***4NT:*** the root directory 570 of the boot drive 557 (***%SystemDrive***\).

This option is most useful if you want to start the program with a specific and unique .INI file 91.

To start the command processor without any .INI file 91, you can create an empty file and specify it as your .INI file 91.

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*

The command processor will skip the leading slash.

//directive=value

This option tells the command processor 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 91, 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 91.

Directives on the command line override any corresponding directive in the .INI file 91.

/D Disable execution of AutoRun commands from Registry. If /D is not specified when ***4NT*** or ***Take Command*** start, they will look for and execute the following registry variables:

HKEY_LOCAL_MACHINE\Software\Microsoft\Command Processor\AutoRun

and / or

HKEY_CURRENT_USER\Software\Microsoft\Command Processor\AutoRun

See also the AutoRun 106 .INI directive.

/H Start ***4NT*** or ***Take Command*** in a hidden window. The window will not appear on the task bar, or in the Alt-tab list of applications.

/L: Forces the use of local lists for aliases, functions directory history and command history, overriding any all LocalAliases=No 130, LocalFunctions=No 131, LocalHistory=No 131, and LocalDirHistory=No 130 directives in the .INI file 91. 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 187, FUNCTION 275, and Local and Global History Lists 52 for more details. Note the required trailing colon (:)!

/LA Forces the use of local aliases. Overrides a LocalAliases=No 130 directive in the .INI file 91.

/LD Forces the use of a local directory history. Overrides a LocalDirHistory=No 130 directive in the .INI file 91.

/LF Forces the use of local functions. Overrides a LocalFunctions=No 131 directive in the .INI file 91.

/LH Forces the use of a local command history list. Overrides a LocalHistory=No 131 directive in the .INI file 91.

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

/S ***(4NT)*** This option tells ***4NT*** that you do not want it to set up a Ctrl-C / Ctrl-Break handler. It is included for compatibility with ***CMD.EXE.***

Warning: It may cause the system to operate incorrectly if you use this option without other software to handle Ctrl-C and Ctrl-Break. This option should be avoided by most users.

/T:*bf* This option sets the foreground and background colors in the command processor command window. Both ***b*** and ***f*** are hexadecimal digits. ***b*** specifies the background color and ***f*** specifies the foreground color. This option is included only for compatibility with ***CMD.EXE.*** See the ***CMD.EXE*** color codes in Colors, Color Names & Codes 552.

In most cases you should set default colors with the StdColors 145 directive in the .INI file 91, or the corresponding **Output Colors** option on the Colors tab 154 of the configuration dialogs 151. If you use both, the /T switch overrides any StdColors 145 directive.

/U This option causes the output of internal commands to a pipe or redirected to a file to be in Unicode when the command processor start. Overrides a UnicodeOutput 148=No directive in the .INI file 91, The command :

OPTION 317 //UnicodeOutput 148=yes | no

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

/V Tells ***4NT / TC*** to handle the ***CMD.EXE*** syntax ***!varname!*** as a delayed expansion of ***%varname***. Since ***CMD.EXE***, unlike ***4NT***/***TC***, doesn't support delayed expansion of variable references in the ***%varname%*** format, it introduced a special ***!varname!*** notation. Using **/V** simply tells ***4NT***/***TC*** to handle that syntax as an alternative to ***%varname%*** or ***%varname*** or ***%[varname]***.

/X This option forces ***4NT*** and ***TC*** 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.EXE***, where it also enables other options. However, in ***4NT*** and ***TC*** 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 the command processor what it must do after startup, and what it should do after completing ***command***. ***Command*** will be executed after the automatic command processor startup program TCSTART ***(TC)*** or 4START ***(4NT)***, 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 the command processor will treat characters after ***command*** as parameters for ***command*** and not as additional startup options.

If ***command*** is preceded by **/C**, the command processor will execute ***command*** and then exit, returning to the parent program or the desktop without displaying a prompt.

The **/K** switch has no effect. Using it is the same as placing ***command*** (with neither **/C** nor **/K**) at the end of the startup command line. It is included only for compatibility with ***CMD.EXE***.

Example 1

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

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

The events below will take place in the order shown:

1 **Windows** starts **c:\4NT\4NT.exe**
2 ***4NT*** initializes from
1st choice: **c:\4NT\4NT.INI**
2nd choice: **c:\4NT.INI**
3.1 If the initialization file was found, **and** it contains the directive
`4StartPath=c:\start`
and one of the files
`c:\start\4start.btm`
`c:\start\4start.bat`
`c:\start\4start.cmd`
`c:\start\4start.exe`
`c:\start\4start.com`

exists, that file is executed by ***4NT***.

3.2 If no initialization file was found in Step **2**, **or**
the initialization file either does not contain the 4StartPath 105 directive, or the value of the directive is ***c:\4NT***, and a **4START** program is found in directory ***c:\4NT***, it is executed by ***4NT***

4 ***4NT*** executes *c:\4NT\start.btm* (or, if not found, it displays an error message).

5 ***4NT*** displays the command prompt, unless an EXIT 262 command was executed in *c:\4NT\start.btm*, terminating ***4NT***.

Example 2

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

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

2.1.2 Automatic Startup and Termination Programs

Command Processor Startup Program

Each time the command processor starts, it looks for a program named 4START 8 (***4NT***) or TCSTART 8 (***TC***). 4START / TCSTART is normally a batch file (*.BAT*, *.BTM*, or *.CMD*), but it can be any executable file. The program must be in the directory specified in the .INI file 91 directive 4StartPath 105 (***4NT***) or TCStartPath 146 (***TC***) if the directive is used. If the directive is not used. the *4START or TCSTART* program, if any, in the same directory as your command processor is executed. Use of 4START / TCSTART is optional, and the command processor will not display an error message if it cannot find the program. If you do not want to use a startup program, either set the StartupFile 144 .INI directive to **NO**, set the start path directive to a directory which does not have one, or leave it unspecified, and make sure that no matching executable file is in the command processor's directory.

4START / TCSTART is a convenient place to change the color or content of the prompt for each session, LOG 303 the start of a session, or execute other special startup or configuration commands. It is also one way to set aliases 187, functions 275, and environment 395 variables. See the section below on Pipes etc. about changing directories via *4START / TCSTART*.

With the exception of some initialization switches 4, the entire startup command line passed to the command processor is available to *4START / TCSTART* as batch file parameters 165 (**%1, %2**, etc.). This can be useful if you want to see the command line passed to the command processor. For example, to pause if any parameters are passed, you could include this command in *4START / TCSTART*:

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

(assuming the command processor's ParameterChar 135 is **$**, the default for ***4NT*** / ***TC***).

Note: ***4NT*** and ***TC*** do not look for or execute *AUTOEXEC.BAT*, but Windows might attempt to parse *AUTOEXEC.BAT* for SET 351 statements at system startup. See your Windows documentation for details.

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

When you set up the *4START / TCSTART* program, remember that it is executed every time the

command processor starts, including when running a pipe 39 or when a transient copy of the command processor is started with the **/C** startup option 4. For example, suppose you enter a command line like this, which uses a pipe:

```
[c:\data] myprog | sort > out.txt
```

Normally this command would create the output file *C:\DATA\OUT.TXT*. However, if your ***4START / TCSTART*** program changes to a different directory, the output file will be written there — not in *C:\DATA*. This is because the command processor starts a second copy (instance) of itself to run the commands on the right hand side of the pipe, and that new copy executes ***4START / TCSTART*** before processing the commands from the pipe.

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

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

Normally this shortcut would LIST 298 the file *C:\DATA\MYFILE.TXT*. However, if *4START / TCSTART* changes the default to a different directory, the command processor. will look for *MYFILE.TXT* there — not in C:\DATA.

Similarly, any changes to environment variables, aliases, or other settings in *4START / TCSTART* will affect all copies of the command processor, including those used for pipes and transient sessions.

You can work around these potential problems with the IF 286 or IFF 287 commands and the _PIPE 420 and _TRANSIENT 423 internal variables. For example, to skip all *4START / TCSTART* processing when running in a pipe or in a transient session, you could use a command like this at the beginning of *4START / TCSTART*:

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

Command Processor Termination Program

Whenever a ***4NT*** or ***TC*** session ends, it looks for a program named 4EXIT 8 (***4NT***) or TCEXIT 8 (***TC***). *4EXIT / TCEXIT* is normally a batch file (*.BAT*, *.BTM*, or *.CMD*), but it can be any executable file. The location of this optional program is determined by the same rule as the location of the *4START / TCSTART* program for the session, and is not necessary in most circumstances. However, it is a convenient place to put commands to save information from one session to another, such as a (command) history list before the command processor exit, or to LOG 303 the end of the session. You can use a termination program even if you have no startup program. You can prevent execution of the termination program by setting the ExitFile 118 .INI directive to NO.

No parameters are passed to the termination program.

2.1.3 Exit Codes

If you start the command processor from another program (*e.g.* to run a batch file or internal command), it will return a numeric code to the other program when it 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.

The command processor's exit code is normally the numeric exit code from the last internal or

external command. However, for compatibility reasons and to avoid conflicts with external commands, only some internal commands set the exit code; others leave it unchanged from the most recent external command.

You can also use the EXIT 262 ***n*** command to explicitly set the exit code. This overrides the rules above, and sets the return code to the parameter of your EXIT 262 command.

2.1.4 Desktop Integration

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

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

Creating Explorer Shortcuts to the Command Processor

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

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

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

For example, it might be useful to keep 4START.BTM 8 very short, and instead rely on global lists (Aliases 187, Functions 275, etc.) after launching SHRALIAS 361 only once, when you first start your system. This can be easily accomplished by creating a batch file (*`startup.btm`*) that loads all your desired aliases, functions and environment variables, handles any housekeeping tasks you wish, then starts SHRALIAS, such as:

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

```
shralias
...
```

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

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

That shortcut could be created with Explorer as outlined above, or with the SHORTCUT 360 command:

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

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

Shortcuts on the desktop can be activated by "double clicking" your pointer device, or - when highlighted - by the **Enter** key on the keyboard. You may also modify the "Properties" of a desktop shortcut by specifying a "shortcut key", which will activate it from your keyboard, regardless of where the pointer cursor is on the screen. Windows XP allows you to specify "Shortcut keys" for "quick launch" icons, but they don't work! **Warning** (Windows bug)**:** if you rename (or move to another directory) a shortcut (link) file which had a shortcut key assigned, that assignment continues to point to the original file, which no longer exists.

Creating Explorer Shortcuts to Batch Files

By default, Windows assumes that **.BAT* and **.CMD* files are to be processed with ***CMD.EXE***, and it knows nothing about **.BTM* files. This means that a shortcut directly pointing to a **.BAT* or **.CMD* file will invoke the "wrong" command processor and that a shortcut pointing directly to a **.BTM* file will generate a "Not a valid WIn32 application" or similar error.

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

- Create shortcuts that invoke the specific command processor you want, e.g. ***d:\path\4nt.exe /c mybatch.cmd***
- Associate filename extensions **.BAT*, **.CMD* and/or **.BTM* with the desired command processor (see Windows File Associations 535, ASSOC 197 and FTYPE 274)

If you don't want to disturb the default associations for **.BAT* and **.CMD* batch files, simply create a new association for **.BTM*, and use that extension for all your own batch files. See also: the Using .BAT Files Under 4NT 163 topic.

Creating Explorer Context Menu Entries

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

While the FTYPE 274 command can be used to change the command associated with the default action, more extensive changes will require the **File Types** Explorer dialog (under the **Tools/Folder Options** menu) or direct modification of the Windows Registry. Select the method with which you are

most comfortable.

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

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

Similar entries can be created at will for any file type. You could add a LIST 298 entry to text files, or **Update JPSTREE Database** to drives, etc.

As usual when dealing with the Registry, use caution and make sure you have a backup of the Registry!

2.2 Directory Navigation

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

With traditional command processors, you change the current drive by typing the new drive letter plus a colon at the prompt. You change the current working directory with the CD 211 command. JP Software command processors support these standard features, and offer a number of enhancements to make directory navigation much simpler and faster.

This section begins with a summary of all the command processor directory navigation features. It also provides detailed documentation on the enhanced directory search features: Extended Directory Searches 14 and the CDPATH 13.

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

Finding Directories

Traditional command processors require you to explicitly type the name of the directory you want to change to. The command processor supports this method, and also offers two significant enhancements:

- The CDPATH 13 variable allows you to enter a specific list of directories to be searched, rather than searching a database. Use CDPATH 13 instead of Extended Directory Searches if you find the extended searches too broad, or your hard drive has too many directories for an efficient search.

- Extended Directory Searches 14 allow the command processor to search a database of all the directories on your system to find the one you want.

Initiating Directory Change

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

- Automatic directory changes 17 allow you to type a directory name at the prompt and switch to it automatically, without typing an explicit CD 211 or similar command.
- The CD 211 command can change directories on a single drive, and can return to the most recently used directory.
- The CDD 213 command changes drive and directory at the same time, and can return to the most recently used drive and directory.
- The PUSHD 331 command changes the drive and directory like CDD 213, and records the previous directory in a directory "stack." You can view the stack with the DIRS 245 command or the @DIRSTACK 446 function, and return to the directory on the top of the stack with POPD 326.

CDD 213, PUSHD 331, and automatic directory changes 17 can also change to network drives and directories mapped to drive letters and to ones specified with UNC names (see File Systems 521 for details).

Returning to a Previous Directory

Traditional command processors do not remember previously-used directories, and can only "return" to a directory by changing back to it with a standard drive change or CD command. JP Software command processors support three additional, simpler methods for returning to a previous directory:

- The CD - 211 and CDD - 213 commands can be used to return to the previous working directory (the one you used immediately before the current directory). Use these commands if you are working in two directories and alternating between them.
- The directory history window 62 allows you to select one of several recently-used directories from a popup list and return to it immediately. The window displays the contents of the directory history list.
- The POPD 326 command returns to the last directory saved by PUSHD 331. The directory stack holds 511 characters, enough for 20 to 40 typical drive and directory entries.

2.2.1 CDPATH feature

When you change directories with an automatic directory change 17 or the CD 211, CDD 213, or PUSHD 331 command, the command processor must find the directory you want to change to. To do so, it first uses the traditional method to find a new directory.

When the traditional search method fails, the command processor tries to find the directory you requested via the CDPATH 13, then via an Extended Directory Search 14. This section covers only CDPATH 13.

Enabling both CDPATH 13 and Extended Directory Searches can yield confusing results, so we recommend that you do not use both features at the same time. If you prefer to explicitly list where the command processor should look for directories, use CDPATH 13. If you prefer to have the command processor look at all of the directory names on your disk, use Extended Directory Searches.

CDPATH 13 is an environment variable, and is similar to the PATH 399 variable used to search for

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

CDPATH is ignored if a complete directory name (one beginning with a backslash \) is specified, or if a drive letter is included in the name. It is only used when a name is given with neither drive letter nor leading backslash.

CDPATH provides a quick way to find commonly used subdirectories in an explicit list of locations. You can create CDPATH with the SET command. The format of CDPATH is similar to that of PATH: a list of directories separated by semicolons ;. For example, if you want the directory change commands to search the *C:\DATA* directory, the *D:\SOFTWARE* directory, and the root directory of drive ***E:*** for the subdirectories that you name, you should create CDPATH with this command:

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

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

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

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

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

or, using an automatic directory change:

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

to change to *D:\SOFTWARE\UTIL*.

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

If you often switch between "sibling" directories, i.e., between subdirectories of a common parent directory. you can enter .. as a search entry in your CDPATH. You can use ... to find "uncles", i.e., a directory one level up (a sibling of the parent directory), thus a subdirectory of the directory 2 levels up.

2.2.2 Extended Directory Searches

When you change directories with an automatic directory change, CD, CDD, or PUSHD command, the command processor must find the directory you want to change to. To do so, it first checks to see whether you have specified either the name of an existing subdirectory below the current directory, or the name of an existing directory with a relative or full path or a drive letter. If you have, the command processor changes to that directory, and does no further searching.

This search method requires that you navigate manually through the directory tree, and type the

entire name of each directory you want to change to. Extended Directory Searches speed up the navigation process dramatically by allowing the command processor to find the directory you want, even if you only enter a small part of its name.

When the first search method fails, the command processor tries to find the directory you requested via the CDPATH 13 variable, then via an Extended Directory Search. This section covers only Extended Directory Searches, which are more flexible and more commonly used than CDPATH. 13

Extended Directory Searches use a database of directory names to facilitate changing to the correct directory. The database is used only if Extended Directory Searches are enabled, and if the explicit directory search and CDPATH 13 search fail to find the directory you requested.

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

You can control the position and size of the popup directory search window from the History tab 155 of the configuration dialogs 151, or with the CDDWinHeight 108, CDDWinLeft 108, CDDWinTop 108, CDDWinWidth 109 directives in the INI file 91. You can also change the keys used in the popup window with key mapping directives 102 in the INI file 91.

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

The Extended Search Database

To create or update the database of directory names, use the CDD /S 213 command. When you create the database with CDD /S, you can specify which drives should be included. If you enable Extended Directory Searches and do not create the database, it will be created automatically the first time it is required, and will include all local hard drives.

The database is stored in the file *JPSTREE.IDX*. By default, the file is placed in the root directory of drive C:.You can specify a different location for this file on the Misc tab 157 of the configuration dialogs 151, or with the TreePath 148 directive in the .INI file 91.

The same tree file is used by all JP Software command processors. If you are using two or more of our products on your computer and want to use different databases for each, provide unique values for the TreePath 148 directive in their respective .INI files 91.

If you use an internal command to create or delete a directory, the directory database is automatically updated to reflect the change to your directory structure. The updates occur if the command processor finds the *JPSTREE.IDX* file in the root directory of drive C: or in the location specified by the TreePath 148 directive.

The TREEEXCLUDE 400 variable can be used to specify which drives/directories should be excluded from inclusion in the directory database.

The internal commands which can modify the directory structure and cause automatic updates of the file are MD 305, RD 333, COPY /S 216, DEL /X 225, MOVE /S 308, and REN 337. The MD 305 **/N** command can be used to create a directory without updating the directory database. This is useful when creating a temporary directory which you do not want to appear in the database.

Enabling Extended Searches

To enable extended directory searches and control their operation, you must set the FuzzyCD 120 directive in the INI file 91. You can set FuzzyCD 120 on the Misc tab 157 of the configuration dialogs 151, or by editing the INI file 91 manually.

- If FuzzyCD 120 = 0, extended searches are disabled, the *JPSTREE.IDX* database is ignored, and CD 211, CDD 213, PUSHD 331 and automatic directory changes search for directories using only explicit names and CDPATH 13. This is the default.

- If FuzzyCD 120 = 1 and an extended search is required, then the command processor will search the *JPSTREE.IDX* database for directory names which exactly match the name you specified.

- If FuzzyCD 120 = 2 and an extended search is required, the command processor will search the database for exact matches first, just as when FuzzyCD 120 = 1. If the requested directory is not found, it will search the database a second time looking for directory names that begin with the name you specified.

- If FuzzyCD 120 = 3 and an extended search is required, the command processor will search the database for exact matches first, just as when FuzzyCD 120 = 1. If the requested directory is not found, it will search the database a second time looking for directory names that contain the name you specified anywhere within them.

For example, suppose that you have a directory called *C:\DATA\MYDIR*, CDPATH 13 is not set, and *C:\DATA* is not the current directory on drive C:. The following chart shows what CDD 213 command you might use to change to this directory.

FuzzyCD 120	Type of extended search	Typical CDD Command
0	CDPATH 13 only (default)	cdd c:\data\mydir
1	CDPATH 13 or exact match	cdd mydir
2	CDPATH 13 or leading match	cdd myd
3	CDPATH 13 or any match	cdd yd

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

Forcing an Extended Search with Wildcards

Normally you type a specific directory name for the command processor to locate, and the search proceeds as described in the preceding sections. However, you can also force the command processor to perform an extended directory search by using wildcard characters 19 in the directory name. If you use a wildcard, an extended search will occur whether or not extended searches have been enabled.

When the command processor is changing directories and it finds *wildcards* in the directory name, it skips the explicit search and CDPATH 13 steps and goes directly to the extended search.

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

Wildcards can only be used in the final directory name in the path (after the last backslash in the path

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

If you use wildcards in the directory name as described here, and the extended directory search database does not exist, it will be built automatically the first time a wildcard is used. You can update the database at any time with CDD 213 **/S**.

Internally, extended directory searches use wildcards to scan the directory database. If FuzzyCD 120 is set to 2, an extended search looks for the name you typed followed by an asterisk (*i.e. DIRNAME**). If FuzzyCD 120 is set to 3, it looks for the name preceded and followed by an asterisk (*i.e. *DIRNAME**).

These internal wildcards will be used in addition to any wildcards you use in the name. For example if you search for *ABC?DEF* (*ABC* followed by any character followed by *DEF*) and FuzzyCD 120 is set to 3, the command processor will search the directory database for **ABC?DEF**.

Disabling Extended Searches in Batch Files

When writing batch files you may want to use the CD 211 or CDD 213 command to switch directories without triggering an extended search. For example, you may need the search to fail (rather than search the extended search database) if a directory does not exist, or you may want to ensure that the extended search popup window does not appear in a batch file designed to run in unattended mode.

To disable extended searches, use the /N option of CD 211 or CDD 213. When this option is used and a directory does not exist below the current directory or on the CDPATH 13, the command will fail with an error message, and will not search the extended search database. For example this command might trigger an extended search:

```
cdd testdir
```

but this one will not:

```
cdd /n testdir
```

Note that this option is not available for PUSHD 331. To perform the same function when using PUSHD 331, save the current directory with PUSHD 331 (without parameters) and then use CDD 213 /N to change directories, for example:

```
pushd
cdd /n testdir
```

2.2.3 Automatic Directory Changes

Automatic directory changes are part of a set of comprehensive directory navigation features built into the command processor. For a summary of these features, and more information on the Extended Directory Searches 14 and CDPATH 13 features mentioned below, see the Directory Navigation 12 section.

The automatic directory change feature lets you change directories quickly from the command prompt, without entering an explicit CD 211 or CDD 213 command. Simply type the name of the directory you want to change to at the prompt, with a terminating backslash (\) (either entered manually, or automatically via the AppendToDir 106 directive). For example:

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

This can make directory changes very simple when it's combined with Extended Directory Searches 14 or CDPATH 13. If you have enabled either of those features, the command processor will use them in searching for any directory you change to with an automatic directory change (see Directory Navigation 12 for more information on CDPATH 13 and Extended Directory Searches 14).

For example, suppose Extended Directory Searches 14 are enabled, and the directory *WIN* exists on drive *E:*. You can change to this directory with a single word on the command line:

```
[c:\tcmd] win\
[e:\win]
```

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

The text before the backslash can include a drive letter, a full path, a partial path, or a UNC name 572 (see File Systems 521 for details on UNC names). Commands like "....\" can be used to move up the directory tree quickly (see Extended Parent Directory Names 62).

If you enter a directory name without the trailing backslash, the parser will change to that directory if no internal or external command of that name is found (and before the UNKNOWN_CMD 195 alias is executed.)

All directory changes, including automatic ones, save the current directory, so it can be recalled with a CDD 213 - or CD 211 - command. For example, any of the following are valid automatic directory change entries:

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

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

2.2.4 Directory Aliases

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

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

You can then reference the files in **c:\program files\jpsoft** by entering **pf:\jpsoft**. Directory aliases work in places that accept filenames and directory names, including filename completion.

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

2.3 File Selection

Most internal commands (like COPY 216, DIR 233, 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 the command processor 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:

2.3.1 Wildcards

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

Wildcards are usually used to specify which files should be processed by a command. If you need to specify which files should not be processed, see File Exclusion Ranges 27 (for internal commands), or EXCEPT 260 (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 19 ***** and the question mark 20 **?**. Additionally, you can specify a set of characters 20. Note the issues about matching short file names 21.

WARNING: When you use a wildcard search for files to process in a command like FOR 267 or DO 246 , and you create new filenames (whether by renaming existing files or by creating new files), the new filenames may match your selection wildcard, and cause you to process them again.

Asterisk * wildcard

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

```
dir *
```

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

```
dir *.txt
```

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

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

The command processor also lets you also use the asterisk to match filenames with specific letters

somewhere inside the name. The following example will display any file with a *.TXT* extension that has the letters ***AM*** together anywhere inside its base name. It will, for example, display *AMPLE.TXT*, *STAMP.TXT*, *CLAM.TXT*, and *AM.TXT*, 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 ***TC*** and ***4NT***.

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

```
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. ***4NT*** and ***TC*** also allow you to specify the exact set of what characters you want to accept (or exclude) in a particular position in the filename by using square brackets **[ ]**. Inside the brackets, you can put the individual acceptable characters or ranges of characters. For example, if you wanted to match *LETTER0.DOC* through *LETTER9.DOC*, you could use this command:

```
dir letter[0-9].doc
```

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

```
dir ?[aeiouy]*.*
```

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

```
dir ?[!aeiouy]*.*
```

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

```
dir [a-dt-v]ip
```

You may use a question mark character inside the brackets, but its meaning is slightly different than a normal (unbracketed) question mark wildcard. A normal question mark wildcard matches any

character, but will be ignored when matching a name shorter than the wildcard specification, as described above. A question mark inside brackets will match any character, but will **not** be discarded when matching shorter filenames. For example:

```
dir letter[?].doc
```

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

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

```
dir letter[].doc
```

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

```
dir /I"[]" *.btm
```

which will display a list of all *.BTM* files which don't have a description, because the empty brackets match only an empty description string (DIR 233 /I selects files to be displayed based on their descriptions).

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

```
dir ??[abc]*[def]*.[pq]*
```

You can also use the square bracket wildcard syntax to work around a conflict between long filenames containing semicolons [**;**], and the use of a semicolon to indicate an include list 30. 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, the command processor will attempt to delete *LETTER1* and then *V2*, because the semicolon indicates an include list 30. 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 directive Win32SFNSearch 149 is set to YES, wildcard searches accept a match on either the LFN or the SFN to match the behavior of CMD.EXE. This may cause some files to be found because of SFN match only. In most situations this is not actually desirable, and can be avoided by setting the directive to NO (the default).

Note: The wildcard expansion process will attempt to allow both CMD.EXE-style "extension" matching (only one extension, at the end of the word) and the advanced ***4NT*** and ***TC*** string matching (allowing things like *.*.abc) when an asterisk is encountered in the destination of a COPY 216, MOVE 308 or REN/RENAME 337 command.

Regular Expressions

You can also use regular expressions for file name tests. (The type of regular expressions to use is specified by the RegularExpressions 140 .INI directive.)

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 526.

2.3.2 Ranges

Most internal commands which accept wild cards also allow size, date, time, exclusion, and description ranges to further define the files that you wish to work with. The command processor 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 **/I**. See Description Ranges 28 for the full syntax.

General Rules

You can reverse the range test by preceding the range argument with the **!** character. For example, to select files that are less than 100 bytes or more than 1000 bytes:

/![s100,1000]

If you combine different types of ranges, a file must satisfy all range specifications to be included. For example,

```
/[d2-8-97,2-9-2005] /[s1024,2048]
```

means files last modified between February 8, 1997 and February 9, 2005, 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, 2005, you could use this command:

```
dir /[d10-1-2005,+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 415 or _TIME 423) and built-in variable functions (*e.g.* @DATE 444, @TIME 492, @MAKEDATE 479, @MAKETIME 479, @FILEDATE 458, @FILETIME 462, or @EVAL 451). 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 24
- Date Ranges 24
- Time Ranges 26
- Exclusion Ranges 27
- Description Ranges 28

Ranges can be used with many commands, including ATTRIB 198, COPY 216, DEL 225, DESCRIBE 230, DIR 233, DO 246, EXCEPT 260, FFIND 262, FOR 267, LIST 298, MOVE 308, RD 333, REN 337, SELECT 345, and TYPE 384

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 32 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 10-27-05 and 10-30-05 will include files

modified on each of those dates, and on the two days in between.

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

2.3.2.1 Size Ranges

Size ranges select files whose size is between the inclusive limits specified. The second parameter of a size range is optional. If you use a single parameter, you will select all files of the specified size or larger. You can also precede the second parameter with a plus sign [**+**]; when you do, it is added to the first value to determine the largest file size to include in the search.

You can exclude a size range by preceding the range with the **!** character.

When you use a size range in a command it should immediately follow the command name. See General Rules for Using Ranges 22 for additional details.

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

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

Examples of size ranges:

Specification	Selects Files of Length
/[s0,0]	zero (empty)
/[s1M]	2**20 bytes or larger
/[s10k,+200]	between 10,000 and 10,200 bytes, inclusive
/[s10,153k]	between 10 and 153,000 bytes, inclusive
/![s1K,5K]	less than 1K or greater than 5K

2.3.2.2 Date Ranges

Date ranges select files dated at any time of day between the inclusive limits specified. For example, **/[d12-1-06,12-5-06]** selects files that were last modified on *or* after December 1, 2006, but not modified *after* December 5, 2006.

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 22 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, **/[d12-31-04,1-1-07]** is equivalent to **/[d12-31-2004,1-1-2007]**, and selects files modified between December 31, 2004 and January 1, 2007.

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 **/[d7-1-05@8:00a,7-3-05@6:00p]** selects files that were modified at any time between 8:00:00 am on July 1, 2005 and 6:00:00 pm on July 3, 2005. 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, the command processor substitutes the current date and time. For example, **/[d10-1-05]** selects files dated between October 1, 2005 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***
/[d1-27-2004,+3]	modified between 1-27-2004 and 1-30-2004
/[d1-27-2004,-3]	modified between 1-24-2004 and 1-27-2004
/[d-0]	modified today (from today minus zero days, to today)
/[d-1]	modified yesterday or today (from today minus one day, to today)
/[d-1,+0]	modified yesterday (from today minus one day, to zero days after that)

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

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

```
dir /[d-3,1/1/80]
```

This reversed date range (with the later date given first) will be handled correctly by the command processor. It takes advantage of the facts that an offset in the start date is relative to today, and that the base or "zero" point for PC file dates is January 1, 1980, or earlier.

You cannot use offsets in the time portion of a date range (the part after an @ sign), but you can combine a time with a date offset. For example, **/[d12-8-05@12:00,+2@12:00]** selects files that were last modified between noon on December 8 and noon on December 10, 2005. 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 26.

- 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, 2005 and February 7, 2005, inclusive, you would use **/[dc2-1-05,2-7-05]**. If you don't specify which date and time to use, the command processor will use the date the file was last modified (written).

NOTE: On 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)

2.3.2.3 Time Ranges

Time ranges select files timed at any time between the two specified times of day. For example, to select files modified at or between noon and 2:00 PM on any day, use **/[t12:00p,2:00p]**. The times in a time range can either be in 12-hour format, with a trailing **a** for AM or **p** for PM, or in 24-hour format.

When you use a time range in a command it should immediately follow the command name. See General Rules for Using Ranges 22 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, the command processor 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 22 for additional details.

Defaults

Start time:	Current time
End time:	Current time
Time type:	Modification (last write)

2.3.2.4 File Exclusion Ranges

Most internal commands which accept wildcards also accept file exclusion ranges to further define the files that you wish to work with. The command processor 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 22 for additional details.

A file exclusion range begins with the switch character (usually a slash), followed by a left square bracket and an exclamation mark **[!** The range ends with a right square bracket **]**.

Inside the brackets, you can list one or more filenames to be excluded from the command. The filenames can include wildcards and extended wildcards 19, 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 22. 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 JP Software's command processors, work for internal commands. The EXCEPT 260 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 246 or FOR 267 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 30.

2.3.2.5 Description Ranges

Most internal commands which accept wildcards also accept description ranges to further define the files that you wish to work with.

When you use a description range in a command it should immediately follow the command name. See General Rules for Using Ranges 22 for additional details.

A description range is specified as **/I"*text*"** where ***text*** is the description to be matched. Wildcards 19 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 526 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 230 for details on file descriptions.

2.3.3 Attribute Switches

Many commands in ***4NT*** and ***TC*** include the **/A:** switch, which allows you to select files for the command to process based on their attributes 524. These switches all use the format **/A[:][-+]RHSAD**. The colon after /A is optional in DIR 233, FFIND 262, and SELECT 345 commands, 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 567 volumes, the extended attributes below are also available.

- **E** Encrypted

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

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

If no attributes are listed at all (*i.e.*, **/A:**), the command will process all files, and (where applicable) all subdirectories, including hidden and system files and directories.

If attributes are combined, all the specified attributes must match for a file to be selected. For example, **/A:RHS** will select only those files with all three attributes set.

If you precede an attribute with a hyphen **-**, files with that attribute will be excluded. For example, **/A:RH-S** selects files which have the read-only and hidden attributes set and which do not have the system attribute set.

If you precede an attribute with a plus **+**, files will be selected which have that attribute turned on or off. When multiple attributes are preceded by **+**, only files which have at least one of these attributes will be selected. For example, **/A:+H+S** will select files with the hidden or system attribute, or both, but will not select files which have neither attribute set. **/A:R+H+S** will select files which are read-only, and also have the hidden or system attribute, or both.

You can combine the plus sign, hyphen, and unmarked attributes to build a specification as complex as you need.

Example

The (dangerous!) command below will make all hidden, 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
```

2.3.4 Multiple Filenames

Most file processing commands can work with multiple files at one time. To use multiple file names, you simply list the files one after another on the command line, separated by spaces. You can use wildcards 19 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 216 or MOVE 308, 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 19 and include lists 30, 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 32 as well as the DO 246, FOR 267 and SELECT 345 commands for other ways of passing multiple file names to a command.

2.3.5 Include Lists

Any internal command that accepts multiple filenames 29 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 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 31 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 216, 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 216 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 233 and SELECT 345 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 345 list. You can see this difference clearly if you experiment with both techniques and the DIR 233 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 19 and multiple filenames 29, include lists work with internal commands, but not with external programs (unless they have been programmed especially to support them).

The maximum length of an include list is 2,047 characters (same as the maximum length of a single file name).

Semicolons in filenames

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

If a filename parameter includes a semicolon, the command processor 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 27.

2.3.6 Delayed Variable Expansion

Some of the internal commands (COPY 216, MOVE 308, PDIR 320, REN 337) 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
```

2.3.7 LFN File Searches

This topic describes special considerations applicable to volumes which support long file names (including VFAT, FAT32, and NTFS volumes). All files on such volumes have a short (FAT-compatible 8.3) file name SFN. A file which was created (or renamed to) a name which contains lower case letters or other characters not compatible with SFN, or a name longer than 8 characters, or an extension longer than 3 characters, or more than one period (**.**) in its name will have both the long file name 567 (LFN) specified, and an SFN 570 automatically generated by the file system. The SFN associated with an LFN may change when the file is moved or copied even when the LFN is not changed.

When ***CMD.EXE*** performs a wildcard search, by default it examines 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, ***4NT*** and ***TC*** default to searching only for the LFN. You can change the default with the Win32SFNSearch 149 .INI directive.

Take extra care when you use wildcards to perform operations on LFN volumes if you have set Win32SFNSearch 149 =Yes, because you may select more files than you intended. For example, Windows often generates short filenames that end ***~1***, ***~2***, etc. If you use a command such as

```
del *1.*
```

you will delete all such files, including most files with long filenames, which is probably not the result you intended!

2.3.8 File Lists

Some 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 298 *@XXX* specifies that LIST 298 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 571 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 233 /B, DIR 233 /F, or FFIND 262, 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 198
COPY 216
DEL / ERASE 225
DESCRIBE 230
DO 246
EXCEPT 260
FOR 267
HEAD 283
LIST 298
MOVE 308
RD / RMDIR 333
REN / RENAME 337
SYNC 369
TAIL 371
TOUCH 381
TYPE 384

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.

2.3.9 Switches for File Selection

Many of the file processing commands (ATTRIB 198, COPY 216, DEL 225, DESCRIBE 230, MOVE 308, REN 337, TYPE 384, 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 22 selections precede the options below in the command line.

The common file selection switches include:

/A:[[-+]rhsadecijopt]	Select files based on their attributes, for example **/A:RH** selects files which have the read-only and hidden attributes set. See Attribute Switches 28 for details; see File Attributes 524 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.

2.4 Executable Extensions

Normally, when you type a filename (as opposed to an alias or internal command name) as the first word on the command line, the command processor looks for a file with that name to execute.

The file's extension may be *.EXE* or *.COM* to indicate that it contains a program; *.PIF* or *.LNK* to indicate that it contains information on how to execute a program under Windows; or *.BTM*, *.BAT*, or *.CMD* to indicate a batch file. 162

You can add to the default list of extensions, and have the command processor take the action you want with files that are not executable programs or batch files. The action taken is always based on the file's extension. For example, you could start your text editor whenever you type the name of a *.DOC* file, or start your database manager whenever you type the name of a *.DAT* file.

Windows also includes the ability to associate file extensions with specific applications. See Windows File Associations 535 for details on this feature and its relationship to executable extensions. See also: Executable Files and File Searches 533.

You use environment variables to define the internal command, external program, batch file, or alias

to run for each defined file extension. To create an executable extension for use only in the command processor, use the SET 351 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, the command processor will search the PATH 399 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
```

the command processor will execute the command:

```
"c:\program files\accessories\wordpad.exe" c:\source\hello.txt
```

Notice that the full pathname of *HELLO.TXT* is automatically included. If you enter parameters on the command line, they are appended to the end of the command. For example, if you changed the above entry to:

```
[c:\source] hello -w
```

the command processor would execute the command:

```
"c:\program files\accessories\wordpad.exe" c:\source\hello.txt -w
```

In order for executable extensions to work, the command, program, batch file, or alias must be able to interpret the command line properly. For example, if a program you want to run doesn't accept a file name on its command line as shown in these examples, then executable extensions won't work with that program.

Executable extensions may include wildcards 19, 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 388 to remove the corresponding variable.

2.5 Input / Output Manipulation

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

Internal commands and some external programs get their input from the computer's standard input device and send their output to the standard output device. Some programs, including ***4NT*** and ***TC***, 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 233 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, 36 Piping, 39 and Keystack. 40 All three are explained in this section. In addition, ***4NT*** and ***TC*** support ANSI X3.64 control sequences in displayed text. The last topic in this section explains ANSI X3.64 support 40 in detail.

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

The TEE 376 and Y 395 commands are "pipe fittings" which add more flexibility to pipes.

2.5.1 Redirection and Piping

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

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

For example, suppose you want a printed list of the files in a directory. If you change the standard output to the printer and issue a DIR 233 command, the task is easy. DIR 233 's output goes to the standard output device, and you have redirected standard output to the printer, so the DIR 233 command prints filenames instead of displaying them on the screen. You can just as easily send the output of DIR 233 (or any other command) to a file or a serial port.

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

The TEE 376 and Y 395 commands are "pipe fittings" which add more flexibility to pipes.

2.5.1.1 Redirection

Redirection can be used to reassign the standard input (stdin 571), standard output (stdout 571), and standard error (stderr 571) devices from their default settings (the keyboard and screen) to another device such as the printer or serial port, to a file, or to the Windows clipboard. You must use some discretion when you use redirection with a device. There is no way to get input from the printer, for example.

Redirection always applies to a specific command, and lasts only for the duration of that command. When the command is finished, the assignments for standard input, standard output, and standard error revert to whatever they were before the command.

In the descriptions below, ***filename*** means either the name of a file or of an appropriate device (**COM *n*** for serial ports; **CON** for the keyboard and screen; **CLIP:** for the clipboard; **NUL** for the "null" device, etc.).

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

Input redirection

< *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 233 command to a file called *DIRLIST*, you could use a command line like this:

```
dir /b *.dat > dirlist
```

You can use any combination of input and output redirection for the same command, as appropriate for your purpose. For example, this command sends input to the external program **SORT** from the file *DIRLIST*, and sends output from **SORT** to the file *DIRLIST.SRT*:

```
sort < dirlist > dirlist.srt
```

You can redirect text to or from the Windows clipboard by using the pseudo-device name **CLIP:** (the colon is required).

If you redirect the output of a single internal command like DIR 233, 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 66 (e.g., ...) ***> report)***, all of the output from the command group is redirected, and redirection ends when the command group is done.

Special considerations for specific commands

You cannot redirect all output from the execution of a DO 246 loop due to the restriction that the DO 246 command and its matching ENDDO 246 may not be part of a command group.

To redirect the output of a TEXT 376 command, append the redirection syntax to the TEXT 376 command.

When you execute a FOR 267 or GLOBAL 279 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 66 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 354 /N1 command, the **Protect redirected output files** setting on the Startup tab 152 of the configuration dialogs, or the NoClobber 133 directive in the INI file 91.

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 NoClobber 133, use ***>!filename***.

Redirection by handle

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

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

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

Handle	Assignment
0	standard input
1	standard output
2	standard error

The ***n>file*** syntax redirects output from handle ***n*** to ***file***. You can use this form to redirect two handles to different places. For example:

```
dir > outfile 2> errfile
```

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

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

```
dir > outfile 2>&1
```

Notice that you can perform the same operations by using standard command processor 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 UNIX-like "here-document" approach. The syntax is:

```
program << word
```

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

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

Special features of "here document":

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

2.5.1.2 Piping

Piping is a special form of redirection, using an additional instance of the command processor 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***

For example, to take the output of the ALIAS 187 command (which displays a list of your aliases and their values) and pipe it to the external SORT utility to generate a sorted list, you would use the command:

```
alias | sort
```

To do the same thing and then pipe the sorted list to the internal LIST 298 command for full-screen viewing:

```
alias | sort | list /s
```

The TEE 376 and Y 395 commands are "pipe fittings" which add more flexibility to pipes.

Like redirection, pipes are fully nestable. For example, you can invoke a batch file and send all of its output to another command with a pipe. A pipe on a command within the batch file will take effect for that command only; when the command is completed, output will revert to the pipe in use for the batch file as a whole. You may also have 2 or more pipes operating simultaneously if, for example, you have the pipes running in different windows or processes.

Processing each line received from a pipe

To process each line of text sent by the left side of a pipe in the command processor, 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: Unlike ***4DOS***, ***4NT*** and ***TC*** implement pipes by starting a new process for the receiving

program. This process goes through the standard secondary shell 535 start-up procedure, including execution of the 4START/TCSTART 8 file, for EACH receiving program. All of the sending and receiving programs run concurrently; the sending program writes to the pipe and the receiving program reads from the pipe. When the receiving program finds an End of File signal, it finishes reading and processing the piped data, and terminates. When you use pipes with ***4NT*** or ***TC***, make sure you consider the possible consequences from using a separate process to run the receiving program, especially that it cannot create/modify/delete environment variables of the sending program, and inclusion of a command to change directories in the 4START/TCSTART 8 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 298, each pipe adds another instance of the command processor.

2.5.2 ANSI X3.64 Support

There is no support for ANSI X3.64 in Windows. For this reason, ***4NT*** and ***TC*** contain their own limited ANSI X3.64 support (key substitutions are not supported). The command processor interprets only its own output, not the output of external applications, except that in ***TC*** a Caveman 86 application using "default colors" will also receive the benefit of ANSI X3.64 support. In some cases you can redirect the output of an application program to a temporary file, then send it through the command processor's ANSI X3.64 interpreter, e.g., by using the TYPE 384 command. This will display ANSI X3.64 correctly, but will not support an interactive application.

To utilize the ***4NT*** or ***TC*** built-in ANSI X3.64 support you must enable it from the Colors tab 154 of the configuration dialogs 151, or with the ANSI 105 directive in the .INI file 91, or with the SETDOS 354 /A command. You can determine whether or not ANSI X3.64 support is enabled with the _ANSI 411 internal variable.

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

For information on the specific ANSI X3.64 commands supported by ***4NT*** and ***TC*** see the ANSI X3.64 Command Reference 550.

2.5.3 Keystack

The KEYSTACK 296 command overcomes two weaknesses of input redirection: *1)* some programs ignore standard input and read the keyboard through the operating system, and *2)* input redirection doesn't end until the program or command terminates. You can't, for example, use redirection to send the first few commands to a program and then type the rest of the commands yourself. But KEYSTACK 296 lets you do exactly that.

KEYSTACK 296 sends keystrokes to an application program. Once the KEYSTACK 296 buffer is empty, the program will receive the rest of its input from the keyboard. KEYSTACK 296 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 296 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 296:

```
keystack F1 Enter
```

See Keys and Key names 550 for details on how key names are entered. See the KEYSTACK 296 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 296 for additional details; see ACTIVATE 186 for information on activating a specific window.

2.5.4 Page and File Prompts

Page Prompts

Several command processor commands can generate prompts, which wait for you to press a key to view a new page or to perform a file activity. When the command processor is displaying information in page mode, for example with a DIR 233 /P or SET 351 /P command, it displays the message

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

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

File Prompts

During file processing, if you have activated prompting with a command such as DEL 225 /P, you will see a prompt similar to the following before processing every file:

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

You can answer this prompt by pressing

Y	Yes	process this file
N	No	do not process this file
A	All	remaining files without further prompting
R	Remaining	files without further prompting

The **R** and **A** responses are equivalent, **A** was added for compatibility with ***CMD.EXE*** versions which display a **Yes/No/All** prompt . You can also press **Esc**, **Ctrl-C**, or **Ctrl-Break** at this prompt to cancel the remainder of the command.

If you press **Ctrl-C** or **Ctrl-Break** while a batch file is running, you will see a **Cancel batch job** prompt. For information on responses to this prompt see Interrupting a Batch File 169.

2.6 Using Internet URLs

If you type an Internet URL (Uniform Resource Locator) which begins with **http:** or **https:** at the prompt, the command processor will pass the URL to Windows. Normally Windows will start your web browser, and request that the browser retrieve the page pointed to by the URL. This feature will only work if Windows can find the proper association between the **http:** or **https:** prefix and the browser software. While this association is standard for most browser installations, it may not be present on all systems.

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

See Waiting for Applications to Finish 68 for information on problems with waiting for the browser to finish after starting a URL.

2.7 Using FTP and HTTP Servers

The command processor allows direct access to remote servers from internal commands such as COPY 216, DEL 225, DIR 233, MOVE 308, MD 305, RD 333, REN 337, and SELECT 345 via several protocols:

- FTP 42 (basic FTP)
- TFTP 45 (Trivial FTP)
- FTPS 45 (SSL FTP)
- HTTP 45 (basic Web access)
- HTTPS 45 (SSL HTTP)

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

- **FTP support:**

The basic filename syntax for anonymous connections is:

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

For example, to get a directory of the JP Software FTP site, you could use this command:

```
Dir ftp://jpsoft.com/*
```

If you don't specify a username and password, ***4NT*** and ***TC*** will look for your FTP user names and passwords in the file *FTP.CFG* (which defaults to the ***4NT*** or ***TC*** directory). You can specify another directory with the FTPCFG 119 directive. You must add entries to the *FTP.CFG* file manually. The format for each line is:

```
url username password [directory template]
```

For example:

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

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

You can also specify an explicit username and password on the command line:

```
ftp://[username:password@]ftp.abc.com/...
```

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

```
dir ftp://username:password@ftp.abc.com/mydir/*
```

You can also use the internal IFTP 288 command to start an FTP session with a server and then use a simplified syntax to manipulate files on the server.

The command processor also supports symbolic hostnames (defined in \windows\system32\drivers\etc\hosts).

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

```
dir ftp://username:password@ftp.abc.com:8765/mydir/*
```

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

```
dir ftp://anonymous:email@domain.com@jpsoft.com/
```

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

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

```
dir ftp:myfile.txt
```

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

The command processor uses standard FTP commands to retrieve information about files and directories and manipulate those files and directories on FTP servers, and relies on the server's compliance with Internet FTP standards. If your server is not fully compliant, or does not operate in the manner that the command processor expects, the results may not be what you intend. For example, if the FTP server you are connecting to is case-sensitive, you may have to use the stored case of file and directory names when you use FTP commands. We urge you to test each server you use with nondestructive commands like DIR before you try to copy or delete files, create or remove directories, etc.

Time-related operations (*e.g.* switches like COPY 216 /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 288 command, the command processor will attempt to re-establish the last connection, if any. That new connection may or may not be logged to the last used directory on that sever. We recommend you always use a full reference (including server name) unless you are specifically taking advantage of an active IFTP 288 connection. You can determine if there is an active IFTP 288 connection with the _iftp 418 and _iftps 418 variables.

Before you can use the built-in FTP support or the IFTP 288 command, you must establish the necessary connection to the Internet. For example, if you use Windows Dial-Up Networking to connect to the Internet, you must start your dial up connection first. If you connect through a proxy server, you must use the Proxy initialization directive 139.

Non-standard FTP servers:

4NT and ***Take Command*** support 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 the command processor to parse the FTP server output. The format is described in the FTP.CFG following the host name, username, and password. The format characters are:

I"text" Do a wildcard comparison of "text" and the directory line; if it matches, discard the entire line. (This is to allow you to skip header & footer lines that would otherwise return garbage.)

"text" Compare (and skip) a literal string (does NOT support wildcard searches)

<space> Skip whitespace (spaces, tabs)

! Skip non-whitespace

- Ignore a single character

F Filename. If the F is followed by a . (i.e., "F."), the extension is the next non-whitespace string. The extension will be appended (preceded by a '.') to the filename.

S Subdirectory flag. If the S is followed by a =, the next character is the character in a "raw" directory listing that denotes a directory. If you don't specify a =, 'D' is assumed.

T Month as a string (i.e., "Jan", "Feb", etc.)

U Unix-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 42 section above for general notes and requirements.

TFTP is only available with COPY 216 (and with MOVE 308 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 42 section above for general notes and requirements.

The **HTTP** syntax is:

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

For example:

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

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

See the FTP 42 section above for general notes and requirements.

The **FTPS** syntax is:

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

For example:

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

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

See the FTP 42 section above for general syntax and requirements.

The **HTTPS** syntax is:

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

For example:

```
copy https://jpsoft.com/xyz.htm
```

2.8 OpenAFS

4NT and ***TC*** have built-in support for OpenAFS. The parser will recognize Unix-style AFS names (i.e., **/afs/athena/user**) and convert them to Windows-compatible names (i.e., **\\afs\athena\user**). (It will also check for custom AFS mount points, and use that name instead of **afs**.)

See http://www.openafs.org for more information on OpenAFS.

3 The Command Line

The command processor displays a **[c:\]** prompt when it is waiting for you to enter a command. The actual text depends on the current drive and directory as well as your PROMPT 329 settings. This is called the command line and the prompt is asking you to enter a command, an alias or batch file name, or the instructions necessary to begin an application program.

This section explains the features that will help you while you are typing in commands, how keystrokes are interpreted when you enter them at the command line, and how to transfer text between the command processor and other applications.

The keystrokes discussed here are the ones normally used by the command processor. If you prefer using different keystrokes to perform these functions, you can assign new ones with key mapping directives 102 in the INI file 91.

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

- Command Line Editing 47
- Command History and Recall 49
- Command History Window 51
- Command Names and Parameters 53
- Filename Completion 58
- Filename Completion Window 61
- Variable Completion 64
- Automatic Directory Changes 17
- Directory History Window 62
- Multiple Commands 65
- Expanding and Disabling Aliases 64
- Command Line Length Limits 71
- Scrolling and History Keystrokes 51
- Command Grouping 66
- Starting Applications 67
- Command Parsing 70
- Date Formats 72

Additional command line features are documented under File Selection 18 and under Directory Navigation 12.

3.1 Command Line Editing

The command line works like a single-line word processor, allowing you to edit any part of the command at any time before you press Enter 118 to execute it, or Esc 117 to erase it.

The command line as typed can contain up to a maximum of 8,191 characters, and it can expand to a maximum of 16,383 characters after variable, function and alias substitution. See Command Line Length Limits 71.

If you do not alter the default key mapping, you can use the following editing keys (among others) when you are typing a command (the words **Ctrl** and **Shift** mean to press the *Ctrl* or *Shift* key together with the other key named). The keystrokes listed here are merely default values, but most editing keys can be redefined via Command Line Editing Keys 103 or General Input Keys 103 directives.

Cursor Movement Keys:

Left 125	Move the cursor left one character.
Right 140	Move the cursor right one character.
Ctrl-Left 150	Move the cursor left one word.
Ctrl-Right 150	Move the cursor right one word.
Home 107	Move the cursor to the beginning of the command.
End 117	Move the cursor to the end of the command.

Insert and Delete Keys:

Ins 124	Toggle between insert and overstrike mode (cursor shape indicates mode).
Del 113	Delete the character under (or to the right of) the cursor, *or* the highlighted text.
Bksp 106	Delete the character to the left of the cursor, or the highlighted text.
Ctrl-L 114	Delete the word or partial word to the left of the cursor.
Ctrl-R 114 or Ctrl-Bksp 114	Delete the word or partial word to the right of the cursor.
Ctrl-Home 114	Delete from the beginning of the line to the cursor.
Ctrl-End 114	Delete from the cursor to the end of the line.
Esc 117	Delete the entire line.
Ctrl-V 135	Paste the first line of text from the clipboard at the current cursor position.
Ctrl-C	***(TC)*** Paste all of the text from the clipboard at the current cursor position.

Highlighting *(4NT)*:

Shift-Right	Highlight character right of cursor and move cursor
Shift-Left	Highlight character left of cursor and move cursor
Shift-Home	Highlight from cursor to beginning-of-line and move cursor
Shift-End	Highlight from cursor to end-of-line and move cursor
Ctrl-Shift-Right	Highlight word right of cursor and move cursor
Ctrl-Shift-Left	Highlight word left of cursor and move cursor
Ctrl-Y 111	Copy highlighted text to the clipboard

Execution:

Ctrl-K 141	Save the current command line in the history list without executing it, and then clear the command line
Ctrl-C or Ctrl-Break	Cancel the command line without saving in the history list. (See also: the CUA 111 directive).
Enter 118	Execute the command line.

Miscellaneous:

F1 284	Get help for the command (first argument on the line)
Ctrl-F1 120	Get help for the current word.
Ctrl-F 105	Expand an alias.
Ctrl-X 149	Expand an environment variable.
Ctrl-A 125	Toggle between LFN and SFN.
Alt-PgUp, Alt-PgDn, Alt-Home, Alt-End, Alt-Up, Alt-Down	***(4NT)*** Scroll the window within the console buffer. (Use the cursor pad keys, not the numeric keypad keys.)

To highlight text on the command line use the mouse ***(TC)***, or hold down the **Shift** key and use any of the cursor movement keys listed above. You can select a complete word by placing the cursor anywhere in the word and double-clicking with the mouse. Once you have selected or highlighted text on the command line, any new text you type will replace the highlighted text. If you press **Bksp** or **Del** while there is text highlighted on the command line, the highlighted text will be deleted.

While you are working at the prompt you can use the clipboard to copy text between the command processor and other applications (see Highlighting and Copying Text 83 for additional details). You can also use Drag and Drop 87 to paste a filename from another application onto the command line.

Most of the command line editing capabilities are also available when the command processor prompts you for a line of input. For example, you can use the command line editing keys when DESCRIBE 230 prompts for a file description, when INPUT 293 prompts for input from an alias or batch file, or when LIST 298 prompts you for a search string.

If you want your input at the command line to be in a different color from the command processor's prompts or output, you can use the Colors tab 154 of the configuration dialogs 151, or the InputColors 124 directive in the .INI file 91.

The command processor will prompt for additional command line text when you include the escape character as the very last character of a typed command line. The default escape character 117 is the caret "**^**", but in general, it's best to use the symbolic "**%=**" EscapeChar 117 representation for portability. For example:

```
echo The quick brown fox jumped over the lazy %=
More? sleeping dog. > alphabet
```

Sometimes you may want to enter one of the command line editing keystrokes on the command line instead of performing the key's usual action. For example, suppose you have a program that requires a Ctrl-R character on its command line. Normally you couldn't type this keystroke at the prompt, because it would be interpreted as a "Delete word right" command. To get around this problem, use the special keystroke Alt-255 110. You enter **Alt-255** by holding down the **Alt** key while you type 0255 on the numeric keypad, then releasing the **Alt** key. This forces the command processor to interpret the next keystroke literally and place it on the command line, ignoring any special meaning it would normally have as a command line editing or history keystroke. You can use **Alt-255** to suppress the normal meaning of command line editing keystrokes even if they have been reassigned with key mapping directives 102 in the .INI file 91, and **Alt-255** itself can be reassigned with the CommandEscape 110 directive.

Alternative Keyboard Input Method:

The method mentioned above for **Alt-255** can be used to generate other characters. You must use the number keys on the numeric keypad, not the row of keys at the top of your keyboard. When this **Alt + keypad** approach is used in a Unicode environment, the command processor will assume that a 3-digit decimal value means an ASCII character, while a 4-digit decimal value mean a Unicode glyph. Make sure that your hardware, character set, code page and font all support the desired combination. Use caution with this method if you plan on manipulating the generated character in other Windows components. See the section on ASCII, Key Codes and ANSI X3.64 Commands 540 for some additional information.

3.2 Command History and Recall

Each time you execute a command, the entire command line is saved in a command history list. You can display the saved commands, search the list, modify commands, and rerun commands. The command history is available at the command prompt and in a special command history window 51. You can choose to use either a local or global command history. 52

Command History Keys:

Up *(4NT)* **Ctrl-Up *(TC)***	Recall the previous (or most recent) command, or the most recent command that matches a partial command line.
Down (*4NT*) **Ctrl-Down *(TC)***	Recall the next (or oldest) command, or the oldest command that matches a partial command line.
F3	Fill in the rest of the command line from the previous command, beginning at the current cursor position.
Ctrl-D	Delete the currently displayed history list entry, erase the command line, and display the previous (matching) history list entry.
Ctrl-E	Display the last entry in the history list.
Ctrl-K	Save the current command line in the history list without executing it, and then clear the command line.
Ctrl-Enter	Copy the current command line to the end of the history list even if it has not been altered, then execute it.
@	As the first character in a line: Do not save the current line in the history list when it is executed, nor store it in the CMDLINE 398 environment variable.

Note: The keystrokes shown above are the default values. See Key Mapping Directives 102 for details on how to assign different keystrokes.

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

```
dir a:*.wks;*.doc
```

to see some of the files on drive A. You might move some new files to drive A and then want to repeat the DIR 233 command. Just press **Up *(4NT)*** or **Ctrl-Up *(TC)*** repeatedly to scan back through the history list. When the DIR 233 command appears, press **Enter** to execute it again. You can also view the command history in a window 51.

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

can also view and manage the command history list with the HISTORY 285 command.

The history list is normally "circular". If you move to the latest command in the list and then press **Up** ***(4NT)*** or **Ctrl-Up** ***(TC)*** once more, you'll see the oldest command in the list. Similarly, if you move to the first command in the list and then press **Up** ***(4NT)*** or **Ctrl-Up** ***(TC)*** once more, you'll see the last command in the list. You can disable this feature and make command history recall stop at the beginning or end of the list by turning off History Wrap on the "History" tab of the configuration dialog, or setting HistWrap 123 to No in the .INI file 91.

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

You can specify the size of the command history list on the **History** tab of the configuration dialog, or with the History 122 directive in the .INI file 91. When the list is full, the oldest commands are discarded to make room for new ones. You can also use the HistMin 122 directive in the INI file 91 to enable or disable history saves and to specify the shortest command line that will be saved.

You can prevent any command line from being saved in the history by beginning it with an at sign (@) or by including it in the
contents of the HistoryExclude 399 variable.

When you execute a command from the history, that command remains in the history list in its original position. The command is not copied to the end of the list (unless you modify it). If you want each command to be copied or moved to the end of the list when it is reexecuted, set HistCopy 121 or HistMove 122 to Yes in your .INI file 91 or select Copy to End or Move to End on the "History" tab of the configuration dialogs. If you select either of these options, the list entry identified as "current" (the entry from which commands are retrieved when you press **Ctrl-Up**) is also adjusted to refer to the end of the history list after each recalled command is executed.

Use **F3** when your new command is different from your previous one by just a character or two at the beginning. For example, suppose you want to execute a `DIR` on several file names then use `DEL` to delete those same files. After the `DIR` is complete type `DEL` and press **F3**; the rest of the command line will be completed for you. Check that it's correct, and then press **Enter** to delete the files. **F3** also retrieves the entire previous command (like **Up** ***(4NT)*** or **Ctrl-Up** ***(TC))*** if nothing has been typed on the line.

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

Use **Ctrl-K** to save some work when you've typed a long command and then realize that you weren't quite ready. For example, if you forget to change directories and notice it after a command is typed or mostly typed, but before you press **Enter**, just press **Ctrl-K** to save the command without executing it. Use the `CD` or `CDD` command to change to the right directory, press **Up** ***(4NT)*** or **Ctrl-Up** ***(TC)*** twice to retrieve the command you saved, make any final changes to it, and press **Enter** to execute it.

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

(TC) If you prefer to use the arrow keys to access the command history in (as in ***4NT***), without having to press **Ctrl**, see the SwapScrollKeys 145 directive, or the corresponding option on the History tab 155 of the configuration dialog. SwapScrollKeys switches the ***TC*** keystroke mapping so that the **Up**, **Down**, **PgDn** and **PgUp** keys manipulate the command history, and **Ctrl-Up**, **Ctrl-Down**, **Ctrl-PgDn** and **Ctrl-PgUp** are used to control the scrollback buffer. For more details see Scrolling and History Keystrokes 51.

3.3 Scrolling & History Keystrokes

In order to support the scrollback buffer 83, some ***TC*** keystroke defaults are different from what you may be used to in ***4NT***. The differences are:

Command Line	*4NT*	*TC*
Previous command	Up	Ctrl-Up
Next command	Down	Ctrl-Down
History window	PgUp	Ctrl-PgUp
Directory history	Ctrl-PgUp, F6	F6

Screen Scrollback	*4NT*	*TC*
Up one line	Alt-Up	Up
Down one line	Alt-Dn	Down
Up one page	Alt-PgUp	PgUp
Down one page	Alt-PgDn	PgDn

If you prefer to use the same keys in ***TC*** as in ***4NT***, use the SwapScrollKeys 145=yes directive in TCMD32.INI 91, or the corresponding option on the History tab 155 of the configuration dialogs 151. SwapScrollKeys 145 switches the keystroke mapping so that the **Up, Down, PgUp** and **PgDn** keys manipulate the command history, and **Ctrl-Up**, **Ctrl-Down**, **Ctrl-PgUp**, and **Ctrl-PgDn** are used to control the scrollback buffer. SwapScrollKeys 145 does not affect the use of **F6** for the directory history.

You can also change the way any individual key operates with the corresponding key mapping directive 102 in TCMD32.INI 91. The directives associated with the history and scrolling keys are:

DirWinOpen 115
HistWinOpen 122
NextHistory 133
PrevHistory 139
ScrollDown 142
ScrollPgDn 142
ScrollPgUp 142
ScrollUp 143

3.4 Command History Window

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

Command History Window Keys:

Up 149	Scroll the display up one line.
Down 116	Scroll the display down one line.
Left 125	Scroll the display left 4 columns.

Right 140	Scroll the display right 4 columns.
PgUp 142	Scroll the display up one page.
PgDn 142	Scroll the display down one page.
Ctrl-PgUp 136 or Home 136	Go to the beginning of the list.
Ctrl-PgDn 137 or End 137	Go to the end of the list.
Ctrl-Enter 137	Move the selected line to the command line for editing
Enter 137	Execute the selected line
Ctrl-D 137	Delete the selected line from the list
Esc 117	Close the window without making a selection.

Note: The keystrokes shown above are the default values. See Key Mapping Directives 102 for details on how to assign different keystrokes.

To activate the command history window press **PgUp** or **PgDn** ***(4NT)*** or **Ctrl-PgUp** or **Ctrl-PgDn** (***TC***) at the command line. A window will appear in the upper right corner of the screen, with the command you most recently executed marked with a highlight. (If you just finished re-executing a command from the history, then the next command in sequence will be highlighted.)

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

You can view a "filtered" history window by typing some characters on the command line, then pressing **PgUp** or **PgDn** ***(4NT)*** or **Ctrl-PgUp** or **Ctrl-PgDn** (***TC***). Only those commands matching the typed characters will be displayed in the window.

You can control the position and size of the history window with configuration directives 99 in the .INI file 91 or the corresponding items on the History tab 155 of the configuration dialogs 151. You can also change the keys used in the window with key mapping directives 102 in the INI file 91.

(***TC***) If you prefer to use the PgUp key to access the command history without having to press **Ctrl** (as in ***4NT***), see the SwapScrollKeys 145 directive in *TCMD32.INI*, or the corresponding option on the History tab 155 of the configuration dialogs 151. SwapScrollKeys 145 switches the keystroke mapping so that the **Down, Up**, and **PgUp** keys manipulate the command history, and **Ctrl-Up**, **Ctrl-Down**, **Ctrl-PgUp**, and **Ctrl-PgDn** are used to control the scrollback buffer. For more details see Scrolling and History Keystrokes 51.

3.5 Local and Global History Lists

The command history 285 and directory history 244 can be stored in either local or global lists.

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

With a global list, all copies of the command processor will share the same history, and any changes made to the history in one copy (e.g., by executing commands from the prompt) will affect all other copies. Global lists are the default.

You can control the type of history list with the LocalHistory 131 and LocalDirHistory 130 directives in the .INI file 91, and with the **/L**, **/LD** and **/LH** options of either the START 364 command or the command processor itself.

If you select a global history list for the command processor, you can share the history among all

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

If you want the histories to be retained in memory even when no command processor session is running, see the SHRALIAS 361 command, which retains the global alias, user-defined function, command history, and directory history lists. SHRALIAS 361 retains the lists in memory, but cannot preserve it when Windows itself is shut down or the user logs out. To save your histories for the next restart of Windows, you must store them in a file and reload them after the system restarts. For details on how to do so, see the HISTORY 285 and DIRHISTORY 244 commands

3.6 Command Names & Parameters

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

```
dir
copy file1 file2 d:\
f:\util\mapmem /v
"c:\program files\JPSoft\4NT\4NT.exe" /LF
```

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

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

If the command name includes a path, the elements must be separated with backslashes (*e.g. F:\UTIL\MAPMEM*). If you are accustomed to Unix syntax where forward slashes are used in command paths, and want the command processor to recognize this approach, you can set the UnixPaths 148 directive to **Yes** in the .INI file 91.

For more information on command entry see Multiple Commands 65 and Command Line Length Limits 71. For details on how the command processor handles the various elements it finds on the command line see Command Parsing 70..

3.7 Conditional Expressions

The commands DO 246 (when used with the UNTIL or WHILE keyword), IF 286, IFF 287/ELSEIFF, and the variable function @IF 469 evaluate a conditional expression, and perform a different action based on whether or not the expression is TRUE. The SWITCH 367 command tests pairs of values for equality. Most of the examples 57 below use the IF 286 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 54
- status test 55
- logical expression 56

Relational Expression

A relational expression compares two character strings, using one of the relational operators 54 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 481 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 443. 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* 54 ***num2***
string1 *relational operator* 54 ***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

Case differences are ignored in string comparisons (except by **EQC**). If two strings begin with the same text but one is shorter, the shorter string is considered to precede (be less than) the longer one. For example, "a" is less than "abc", and "hello_there" is greater than "hello".

When you compare text strings, you may need to enclose the parameters in double quotes in order to avoid syntax errors which can occur if one of the parameter values is empty (*e.g.*, due to an environment variable which has never been assigned a value). This technique will not work for numeric comparisons, as the quotes will force a string comparison, so with numeric tests you must be sure that all variables are assigned values before the test is done.

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

CMD.EXE	**4NT** and **TC**
EQL or **EQU**	**EQ**
NEQ	**NE**
LSS	**LT**
LEQ	**LE**
GTR	**GT**
GEQ	**GE**

Internal variables 402 and variable functions 424 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 command processor status. In addition to the tests below, there are many internal variables and variable functions which allow you to test the status of many other parts of the system.

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

DEFINED *variable* If variable exists in the environment, the expression is true. This is equivalent to testing whether or not variable is nonempty.

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

Note: GOSUB variables 280 and internal variables 402 always fail the DEFINED test.

ERRORLEVEL [relational operator 54] 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 54 may be any of those listed above (e.g., EQ, GT). If no operator is specified, the default is GE. The comparison is done numerically.

Not all programs return an explicit exit code. For programs which do not, the behavior of ERRORLEVEL is undefined.

EXIST *filename*	If filename matches a file which exists, the expression is true. You can use wildcards in filename, in which case the expression is true if any file matching the wildcard name exists. filename may include an absolute or relative path. WARNING: In Windows the expression will be true if there is either a file or a directory named filename. Use ISFILE or ISDIR instead. The special filename NUL is commonly used in CMD.EXE batch files to test the existence of a directory. The expression exist xxx\NUL is true only if xxx is a directory.
ISALIAS *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 31 for details on the correspondence between short and long filenames). This test may require DEBUG privilege.
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 354 /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.
ISPLUGIN *name*	If name is a plugin 325 variable, function, or command, the expression is true.
ISWINDOW "title"	If a window which matches the title exists, the expression is true. double quotes must be used around the title, which may contain wildcards and extended wildcards.
PLUGIN *module*	If the plugin module is loaded, the expression is true.

Logical Expressions

A logical expression is one of the following:

- a relational expression 54
- a status test 55

- the unary logical operator **NOT** followed by a logical expression
- two logical expressions connected by a binary logical operator

Logical operators

operator	type	usage	value is TRUE if
NOT	unary	**NOT** *cond*	*cond* is FALSE.
.AND.	binary	*cond1* **.AND.** *cond2*	both *cond1* and *cond2* are TRUE.
.OR.	binary	*cond1* **.OR.** *cond2*	at least one of *cond1* and *cond2* is TRUE.
.XOR.	binary	*cond1* **.XOR.** *cond2*	one of *cond1* and *cond2* is TRUE, and the other one is FALSE.

This example runs a program called ***DATALOAD*** if today is Monday or Tuesday (enter this on one line):

```
if "%_dow" == "Mon" .or. "%_dow" == "Tue" dataload
```

Test conditions are always scanned from left to right – there is no implied order of precedence, as there is in some programming languages. You can, however, force a specific order of testing by grouping conditions with parentheses, for example (enter this on one line):

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

Combining logical expressions

Parentheses can be used only when the portion of the ***expression*** inside the parentheses contains at least one of the binary logical operators **.and.**, **.or.**, or **.xor.**. Parentheses on a simple expression which does not combine two or more tests will be taken as part of the string to be tested, and will probably make the test fail. For example, the first of these tests is **FALSE**, the second is **TRUE**:

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

Parentheses may be nested.

Examples

This batch file fragment runs a program called *WEEKLY* if today is Monday:

```
if "%_dow" == "mon" weekly
```

This batch file fragment tests for a string value:

```
input "Enter your selection : " %%cmd
if "%cmd" == "WP" goto wordproc
if "%cmd" NE "GRAPHICS" goto badentry
```

This example calls *GO.BTM* if the first two characters in the file *MYFILE* are **GO**:

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

The first batch file fragment below tests for the existence of *A:\JAN.DOC* before copying it to drive **C** (this avoids an error message if the file does not exist):

```
if isfile a:\jan.doc copy a:\jan.doc c:\
```

This example tests the exit code of the previous program and stops all batch file processing if an error occurred:

```
if errorlevel == 0 goto success
echo "External Error – Batch File Ends!"
cancel
```

3.8 Filename Completion

Filename completion can help you by filling in a complete file name on the command line when you only remember or want to type part of it. Filename completion can be used at the command line, which is explained here, and in a filename completion window 61.

Filename Completion Keys:

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

Note: The keystrokes shown above are the default values. See Key Mapping Directives 102 for details on how to assign different keystrokes.

For example, if you know the name of a file begins *AU* but you can't remember the rest of the name, type:

```
copy au
```

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

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

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

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

The pattern you use for matching may contain any valid filename characters, as well as wildcard characters and extended wildcards 19. For example, you can copy the first matching *.TXT* file by typing

```
copy *.txt
```

and then pressing **Tab**.

If you don't specify part of a filename before pressing **Tab**, the command processor will match all files. For example, if you enter the above command as "**COPY** ", without the **.TXT*, and then press **Tab**, the first filename in the current directory is displayed. Each time you press **Tab** or **F9** after that, another name from the current directory is displayed, until all filenames have been displayed. ***Note:*** you must terminate the command (e.g., by space) before file completion becomes available.

If you type a filename without an extension, the command processor will append ***** to the name on LFN 567 drives, and ***.*** on drives which only support short file names 570. It will also place a ***** after a partial extension. If you are typing a group of file names in an include list 30, the part of the include list at the cursor will be used as the pattern to match.

When filename completion is used at the start of the command line, it will only match directories, executable files, and files with executable extensions 33, since these are the only file names that it makes sense to use at the start of a command. If a directory is found, a \ will be appended to it to enable an automatic directory change 17. If you need to complete the name of any other file at the start of the command line, press **Space** before starting to type the name. Filename completion will then match any name, not just directory and executable names. *Note* that you can also "execute" files whose extension has an association in the Windows Registry, but such files are not considered executable by the command processor, and only the method above using a space will work.

Filename completion occurs in the physical order in which matching filenames are stored in the directory, the same order in which DIR 233 /O:U would list them. That order is determined by the underlying file system.

The command processor also supports network server and sharename completion. If the filename begins with \\, the completion routines will enumerate the network resources for matching server and/or share names. You can control the way server name completion functions with the ServerCompletion 143 directive in the .INI file 91.

Filename completion will search the PATH 319 for an executable filename if you have set the CompletePaths 110 directive, and you are :

(1) at the beginning of the command line, and
(2) there are no matching entries in the current directory, and
(3) the name you are attempting to match doesn't contain a full or partial path specification.

If all three conditions are met, filename completion will return the first matching executable found in the PATH 319.

If you are on an NTFS drive, you can also complete stream names. For example:

```
copy test:t
```

and then pressing **Tab** will search the file ***test*** for streams beginning with "t". Note that you cannot complete a filename and a stream name simultaneously (i.e., **t*:t***).

Several topics are related to filename completion. See:

- Converting Between Long and Short Filenames 61
- Appending Backslashes to Directory Names 62
- Customizing Filename Completion 60
- Filename Completion Window 61

- Variable Name Completion 64

3.9 Customizing Filename Completion

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

To customize filename completion you can use Editing tab 154 of the configuration dialogs, or set the FileCompletion 118 directive manually in the .INI file 91. You can also use the FILECOMPLETION 399 environment variable. If you use both, the environment variable will override the settings made in the dialog or the INI file 91. You may find it useful to use the environment variable for experimenting, then create permanent settings with the configuration dialog or the FileCompletion directive.

The format for both the environment variable and the directive is:

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

where

cmd1 etc. are command names

ext1 etc. are file extensions (which may include wildcards) or one of the following file types:

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

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

The command name is the internal command, alias, or executable file name (without a path). For example, to have file completion return only directories for the CD 211 command and only *.C* and *.ASM* files for a Windows editor called WinEdit, you would use this setting for filename completion in the configuration dialog:

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

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

```
set filecompletion=cd:dirs; winedit:c asm
```

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

When testing to see if customized filename completion should be used, the command processor checks the actual command line you type (but without expanding any aliases). For example, if you use the FileCompletion 399 setting shown above and have "W" aliased to "WINEDIT," and then enter a "W" command, the FileCompletion 399 setting – which refers only to "WINEDIT" – will be ignored.

To use customized filename completion for aliases you must enter the alias name in the FileCompletion 399 setting:

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

3.10 Filename Completion Window

You can view matching filenames in a filename completion window. To activate the window, press **F7** or **Ctrl-Tab** at the command line. You will see a window in the upper-right corner of the screen, with a sorted list of files that match any partial filename you have entered on the command line. If you haven't yet entered a file name, the window will contain the name of all files in the current directory. You can search for a name by typing the first few characters. See Popup Windows 536 for details.

Filename Completion Window Keys:

F7 or Ctrl-Tab 136	(from the command line) Open the window.
Up 149	Scroll the display up one line.
Down 116	Scroll the display down one line.
Left 125	Scroll the display left 4 columns.
Right 140	Scroll the display right 4 columns.
PgUp 142	Scroll the display up one page.
PgDn 142	Scroll the display down one page.
Ctrl-PgUp 136 or Home 136	Go to the beginning of the list.
Ctrl-PgDn 137 or End 137	Go to the end of the list.
Enter 137 or Double Click	Insert the selected filename into the command line.

Note: The keystrokes shown above are the default values. See Key Mapping Directives 102 for details on how to assign different keystrokes.

See also: Filename Completion 58

3.11 Converting Between Long & Short Filenames

On LFN drives, the command processor will search for and display long filenames during filename completion. If you want to search for 8.3 short filenames, press Ctrl-A 125 before you start using filename completion. This allows you to use filename completion on LFN drives with applications that do not support long filenames. The LFNToggle 125 directive can be used to change the keystroke assigned to this feature.

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

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

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

3.12 Appending Backslashes to Directory Names

If you set the AppendToDir 106 directive in the INI file 91 to **Yes**, or the corresponding option in the configuration dialogs, the command processor will add a trailing backslash **** to directory names. The character appended is a slash **/** for directory names in FTP URLs 42 (or if you have set the directive UnixPaths=yes) 148 to all directory names.

This feature can be especially handy if you use filename completion to specify files that are not in the current directory — a succession of **Tab** or **F9** and **F10** ***(4NT)*** or **Ctrl-Shift-Tab** ***(TC)*** keystrokes can build a complete path to the file you want to work with.

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

```
1  edit \da <F9>
2  edit \data\
3  edit \data\f <F9>
4  edit \data\frank.doc <F9>
5  edit \data\finance\
6  edit \data\finance\map <F9>
7  edit \data\finance\maps.dat
```

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

3.13 Extended Parent Directory Names

The command processor allows you to extend the syntax for naming the parent directory, by adding additional **.** characters. Each additional **.** represents an additional directory level above the current directory. For example, *.\FILE.DAT* refers to a file in the current directory, *..\FILE.DAT* refers to a file one level up, i.e., in the parent directory, and *...\FILE.DAT* refers to a file two levels up, i.e., in the parent of the parent directory. If 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 466 function to expand file and directory references when necessary:

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

3.14 Directory History Window

[The directory history window is part of a set of comprehensive directory navigation features built into the command processor. For a summary of these features, and more information on enhanced

directory navigation features, see Directory Navigation 12.]

Directory History Window Keys:

F6 115	Open the window from the command line (***TC***)
Ctrl-PgUp 115	Open the window from the command line (***4NT***)
Up 149	Scroll the display up one line.
Down 116	Scroll the display down one line.
Left 125	Scroll the display left 4 columns.
Right 140	Scroll the display right 4 columns.
PgUp 142	Scroll the display up one page.
PgDn 142	Scroll the display down one page.
Ctrl-PgUp 136 or Home 136	Go to the beginning of the list.
Ctrl-PgDn 137 or End 137	Go to the end of the list.
Ctrl-Enter 137	Move the selected line to the command line for editing
Enter 137	Change to the selected drive/directory
Ctrl-D 137	Delete the selected line from the list
Esc 117	Close the window without making a selection.

Note: The keystrokes shown above are the default values. See Key Mapping Directives 102 for details on how to assign different keystrokes.

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

You can view the directory history from the scrollable directory history window and change to any drive and directory on the list. To activate the directory history window, press F6 115 at the command line. You can then select a new directory with the Enter 137 key or by double-clicking with the mouse.

If the directory history list becomes full, old entries are deleted to make room for new ones. You can set the size of the list with the DirHistory 115 directive in the .INI file 91 or with the corresponding items on the History tab 155 of the configuration dialogs 151. You can change the keys used in the window with key mapping directives 102 in the INI file 91.

In order to conserve space, each directory name is recorded just once in the directory history, even if you move into and out of that directory several times. The directory history can be stored in either a local or global list; see below for details.

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

You can also view and manage the directory history list with the DIRHISTORY 244 command.

Local and Global Directory History

The directory history can be stored in either a local or global list. With a local directory history list, any changes made to the list will only be known only to the current copy of the command processor. They will not be visible in other sessions. Whenever you start a secondary shell which uses a local history list, it inherits a copy of the directory history from the previous shell. However, any changes to the history made in the secondary shell will affect only that shell.

All copies of the command processor using global directory history list will share a single copy of directory history. Any directory changes made in any of these copies of the command processor will

be recorded in that shared list, and be accessible by all of them. However, any additional copies of the command processor which use local directory history will see their own local lists. Global lists are the default for ***4NT*** and ***TC***.

You can control the type of history list from the Startup tab 152 of the configuration dialogs, with the LocalDirHistory 130 directive in the .INI file 91, with the /L and /LD command line options 4, and with the /L and /LD options of the START 364 command.

When you close all command processor sessions, the memory for the global directory history list is released, and a new, empty directory history list is created the next time you start the command processor. If you want the directory history list to be retained in memory even when no copy of the command processor is running, you need to execute the SHRALIAS 361 command, which performs this service for the global command history, directory history, user-defined functions, and aliases.

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

3.15 Variable Name Completion

Variable name completion works like filename completion. 58 If the parameter begins with a %, the completion routines will scan the environment for matching variable names. For example, if the PROMPT 400 and PATH 399 variables are in the environment, in that order, and no other variables start with p, the sequence below may be used to display the value of PATH 399:

```
echo %p<Tab>
echo %PROMPT<Tab>
echo %PATH<Enter>
```

Note: Variable name completion does not work for variable functions.

3.16 Expanding and Disabling Aliases

A few command line options are specifically related to aliases, and are documented briefly here for completeness. If you are not familiar with aliases, see Aliases 160 and the ALIAS 187 command for complete details.

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

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

```
*dir
```

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

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

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

will endlessly try to redefine itself.

3.17 Multiple Commands

At times, you probably know the next two or three commands that you want to execute. Instead of waiting for each one to finish before you type the next, you can type them all on the same command line, separated by the command separator 559 (by default, an ampersand **&**) or the **%+** pseudovariable reference. For example, if you know you want to copy all of your *.TXT* files to drive **A:** and then run CHKDSK to be sure that drive A's file structure is in good shape, you could enter the following command:

```
copy *.txt a: %+ chkdsk a:
```

You may put as many commands on the command line as you wish, as long as the total length of the command line does not exceed 8,191 characters.

You can use multiple commands in alias 187 definitions and batch files 162 as well as from the command line.

If you don't like using the default command separator, you can pick another character using the SETDOS 354 command's /C option, the CommandSep 110 directive in the .INI file 91 or on the Syntax tab 156 of the configuration dialogs 151.

3.18 Conditional Commands

When an internal command or external program finishes, it returns a result called the exit code 9. Conditional commands allow you to perform tasks based upon the previous command's exit code 9. 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 9 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 9 is non-zero. For example, if the following BACKUP operation fails, then ECHO 253 will display a message:

```
backup c:\ a: || echo Error in the backup!
```

All internal commands return an exit code 9, 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 9. To determine whether a particular external program returns a meaningful exit code 9 use an ECHO %? command immediately after the program is finished. If the program's documentation does not discuss exit code 9, you may need to experiment with a variety of conditions to see how the exit code 9 changes.

3.19 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 337 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 337 commands enclosed in the parentheses appear to GLOBAL 279 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 260, FOR 267, GLOBAL 279, and IF 286 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 were typed at the command line, then the command processor would issue a **More?** prompt in response to each line until the command group is closed (i.e. the final parenthesis is recognized) as discussed below.

The second common use of command grouping is to redirect input or output for several commands without repeatedly using the redirection 36 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 253 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 36 and piping 39.

You can also use command grouping in a batch file or at the prompt to split commands over several lines. This last example is like the redirection example above, but is entered at the prompt. Note the **More?** prompt after each incomplete line. None of the commands are executed until the command group is completed with the closing parenthesis. This example does not have to be entered on one line:

```
[c:\] (echo Data files %_date
More? dir *.dat
More? echo.
More? echo Text files %_date
More? dir *.txt) > filelist
[c:\]
```

Limitations

A group of commands in parentheses is like a long command line. The total length of the group may not exceed 8,191 characters, whether the commands are entered from the prompt, an alias, or a batch file. The limit includes the space required to expand aliases and environment variables used within the group.

Each line you type at the normal prompt or the **More?** prompt, and each individual command within the line, must be within the usual command line length limit 71.

3.20 Starting Applications

The command processor offers several ways to start applications.

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

The command processor offers two methods to simplify and speed up access to your applications. One is to create an alias 187, for example:

```
alias myapp d:\apps\myapp.exe
```

In ***Take Command*** you can also use the Tool Bar to start frequently used applications. For example, a tool bar button named **MyApp** which invokes the command *d:\apps\myapp.exe* would accomplish the same thing as the alias shown above. You can use these methods together. For example, if you define the alias shown above you can set up a tool bar button called **MyApp** and simply use the command **myapp** for the button, which would then invoke the previously-defined alias.

You can also start an application by typing the name of a data file associated with the application. The command processor will examine the file's extension and run the appropriate application, based on executable extensions 33 or Windows file associations 535.

For additional flexibility, you can also start applications with the START 364 command. START 364 provides a number of switches to customize the way an application is started.

Searching for Applications

When you start an application without specifying a path, the command processor searches for the application in the current directory, and then all directories on the PATH. The command processor also searches the **Windows** and **Windows system** directories; see the PATH 319 command for details. (If you do enter an explicit path, the command processor will only look in the directory you specified.)

If you enter a file name with no extension, the command processor will search each directory for a matching *.PIF*, *.COM*, *.EXE*, *.BTM*, *.BAT*, or *.CMD* file (and *.REX* and/or *.REXX* if a REXX interpreter is loaded), then for a file matching a Windows file association or executable extension. That search order may be altered via the advanced PathExt 135 directive. If no such file is found, the command processor will move on to the next directory in the search sequence.

***(TC)* Application Windows**

In most cases, ***TC*** starts each application in its own window. When the application exits, the window is closed. However, ***TC*** runs character mode applications in a *console window* associated with the ***TC*** process. This window is opened automatically, and remains open as long as ***TC*** is running. The window is visible whenever an application is actually running within it. After an application exits, you can switch back to the console window and view the output with the **Alt-V** key, or the View Console Window selection on the **Apps** menu. Applications run under Take Command's Caveman 86 feature will run in the console window, but their output will be displayed in the ***TC*** window. For additional details, see Console Applications and the Console Window 86.

3.21 Waiting for Applications to Finish

When you start a Windows application from the prompt, the command processor does not normally wait for the application to finish before returning to the prompt . This allows you to continue your work at the prompt while the application is running. You can force the command processor to wait for applications to finish before continuing by selecting the **Wait for External Apps** checkbox on the Startup tab 152 of the configuration dialog, with the ExecWait 118 directive in the .INI file 91, or with the START 364 command's /WAIT switch (START 364 can also control many other aspects of how your applications are started).

Regardless of the ExecWait 118 setting, the command processor always waits for applications that are run from transient shells (with a /C), or from batch files before continuing with subsequent commands in the batch file. To start an application from a transient shell or a batch file and continue without waiting for the application to finish, use the START 364 command (without the /WAIT switch).

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

3.22 Escape Character

The command processor recognizes a user-definable escape character. This character gives the character that follows a special meaning; it has a different purpose than the ASCII **ESC** that is often used in ANSI X3.64 and printer control sequences.

The default escape character is a caret (**^**, ASCII: 94). If you don't like the default escape character, you can pick another character using the SETDOS 354 /E command, the configuration dialogs, or the EscapeChar 117 directive in your .INI file 91. If you plan to share aliases or batch files between several ***4NT*** and ***TC*** configurations, use the %= 410 pseudovariable, which is accepted in all of them, regardless of the actual value assigned to the escape character. See the section on Special Character Compatibility 72 for details about choosing compatible escape characters for two or more products. Note that if you change the default, your batch files will not work under ***CMD.EXE*** and you won't be able to run third-party batch files.

Ten special characters are recognized when they are preceded by the escape character. The combination of the escape character and one of these characters is translated to a single character, as shown below. These are primarily useful for redirecting 36 codes to the printer. The special characters which can follow the escape character are:

Codes for Escape Characters

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

If you follow the escape character with any other character, the escape character is removed and the second character is copied directly into the command line. This allows you to suppress the normal meaning of special characters (such as **? * / \ | " ` > <** and **&**). For example, to display a message containing a **>** symbol, which normally indicates redirection:

```
echo 2 is %=> 4
```

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

WARNING: Escape characters are considered to be normal characters on the right side of a pipe, e.g., in the command below:

```
type f:zonk|grep32 /zq ^^d
```

Note: The term **escape character** has two additional usages not related to the above description, as detailed in the description of the PROMPT 329 command and in ASCII, Key Codes and Key Names 540 .

3.23 Command Parsing

Whenever you type something at the command line and press the Enter 118 key, or include a command in a batch file, you pass a command to the command processor, 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.

The command processor goes through several steps when parsing a command line. Before it starts, it writes the entire command line (which may contain multiple commands 65) to the history log file if history logging has been enabled (with the LOG /H 303 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

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

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

The command name is **dir**, and the command tail is "***.txt /2/p/v**". In some instances, the parser will be able to understand incorrect syntax such as **dir/w**, but there should always be at least one space between the command name and its parameters.

2. Expanding aliases

Next, the command processor tries to match the command name against its list of aliases 160. If it finds a match between the command name and one of the aliases you've defined, it replaces the command name with the contents of the alias. This substitution is done internally and is not normally visible to you; however, you can view a command line with aliases expanded by pressing Ctrl-F 105 after entering the command at the prompt.

If the alias included parameters (%1, %2, etc.), the parameter values are filled in from the text on the command line, and any parameters used in this process are removed from the command line. The process of replacing a command name that refers to an alias with the contents of the alias, and filling in the alias parameters, is called alias expansion.

This expansion of an alias creates a new command name: the first word of the alias. This new command name is again tested against the list of aliases, and if a match is found the contents of the new alias is expanded just like the first alias. This process, called nested alias expansion, continues until the command name no longer refers to an alias.

3. Expanding variables

The next step is to locate any batch file or alias parameters, environment variables, internal variables, or variable functions in the command, and replace each one with its value (see " Environment: Variables and Functions 395"). This process is called variable expansion, and is not normally visible. However, you can view an expanded command line by pressing Ctrl-X 149 after entering the command at the prompt.

The variable expansion process is modified for certain internal commands, such as EXCEPT 260, IF 286, and GLOBAL 279. 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, the command processor next tries to match the resulting command name with its list of plugin 325 commands or internal commands 176. If it is unsuccessful, it knows that it will have to search for a batch file or external program to execute your command.

5. Displaying the command

When all of the aliases and environment variables have been expanded, the command processor will echo the complete command to the screen (if command line echo has been enabled) and write it to the log file 131 (if command logging 303 has been turned on).

6. Processing redirection and piping

Before it can actually execute your command, the command processor must scan the command tail to see if it includes redirection 36 or piping 39. 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 69 are processed. However, this might also already have taken place inside some of the variable functions (such as @IF 469) that are likely to pass escaped strings in their parameters. If you are referencing one of those in an ECHO 253 or similar command, you need to escape twice ("^^") or use SETDOS /X 354 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. The command processor will first look for a matching plugin 325 command name; if it doesn't exist then it tries to match an internal command. Otherwise, the command processor searches for an executable (*.COM* or *.EXE*) file, a batch file, or a file with an executable extension that matches the command name (see the detailed description of this search in Executable Files and File Searches 533).

9. Cleaning up

Once the internal command or external program has terminated, the command processor saves the result or exit code that the command generated, cleans up any redirection that you specified, and then returns to the original command line to retrieve the next command. When all of the commands in a command line are finished, the next line is read from the current batch file, or if no batch file is active, the prompt is displayed.

Note: You can disable and reenable several parts of command parsing (for example alias expansion, variable expansion, and redirection) with the SETDOS /X 354 command.

3.24 Command Line Length Limits

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

As the command processor scans the command line and substitutes the contents of aliases user

defined functions, and environment variables for their names, the line usually gets longer. This expanded line is limited to 16,383 characters. If your use of aliases, user defined functions, or environment variables causes the command line to exceed the applicable one of these limits as it is expanded, you will see a **Command line too long** error and the remainder of the line will not be executed.

3.25 Special Character Compatibility

If you want to share aliases, user defined functions, and batch files with other users, you need to be aware of possible differences in three important characters: the Command Separator (see Multiple Commands 65), the Escape Character (see Escape Character 69), and the Parameter Character (see Batch File Parameters 165).

The default values of each of these characters in each product is shown in the following chart.

Product	Separator	Escape	Parameter
4DOS (obsolete)	^	Ctrl-X	&
4NT / TC	&	^	$
pseudovariable	**%+**	**%=**	

In your batch files and aliases, and even at the command line, you can smooth over these differences in three ways:

1. **Preferred**: use internal pseudovariables that contain the current special character, rather than using the character itself (see %+ 410 and %= 410). For example, this command:

   ```
   if "%1" == "" (echo Parameter missing! ^ quit)
   ```

   will only work if the command separator is a caret. However, this version works regardless of the current command separator:

   ```
   if "%1" == "" (echo Parameter missing! %+ quit)
   ```

2. Select a consistent set of characters from the Syntax tab 156 of the configuration dialogs 151, or with configuration directives 99 in the .INI file 91.

3. In a batch file, use the SETLOCAL 358 command to save the command separator, escape character, and parameter character when the batch file starts. Then use SETDOS 354 as described below to select the characters you want to use within the batch file. Use an ENDLOCAL 256 command at the end of the batch file to restore the previous settings.

You can also use the SETDOS 354 command to change special characters on the command line.

3.26 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 233 command and functions returning a date without a format code specifier. The year can be entered as a 4-digit or 2-digit value. Two-digit years from 80 to 99 are interpreted as 1980...1999; values from 0 to 79 are interpreted as 2000...2079. Month and day may be entered without a leading zero. Most non-numeric printing characters are accepted as field separators. All three fields must be specified.

3.27 Case Sensitivity

With the following exceptions, the command processor treats upper case and lower case letters identically:

The relational operator **EQC**
The character manipulation functions **@ascii**, **@unicode**, **@repeat**, **@replace**, **@strip** and **@wild**.

The codes used to specify units of storage size (**kKmMgGtT**) in:

- size ranges
- disk space and file size reporting functions

4 Take Command Interface

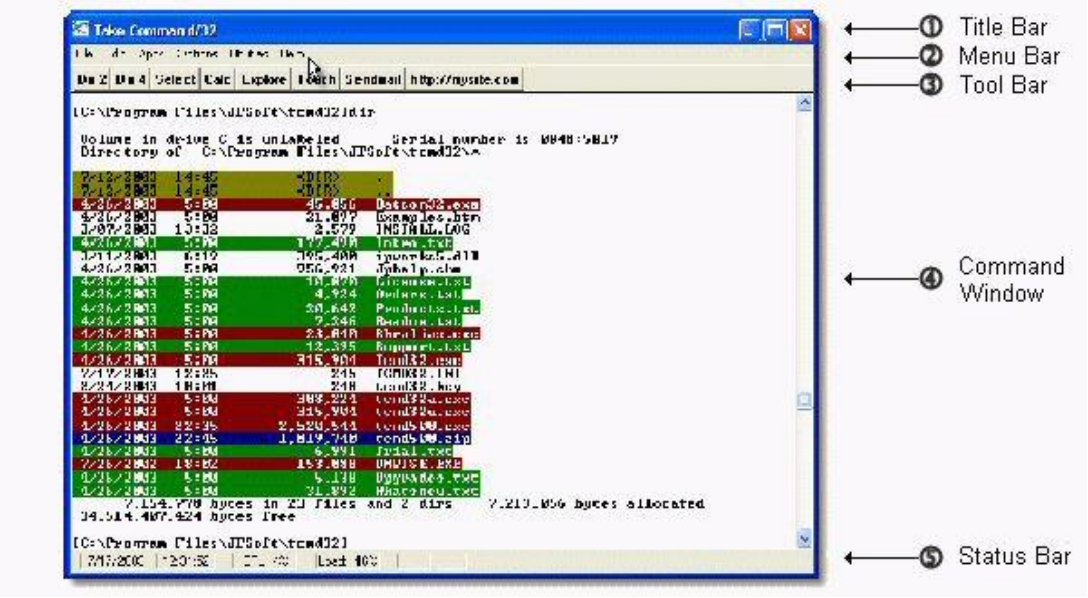
The *Take Command* Window

Starting Applications

***Take Command* and the Windows Environment**

4.1 The Take Command Window

The ***TC*** window has **five** parts:



1. The **Title Bar** is the same as the one used in most Windows applications, with a control menu button on the left and the minimize, maximize, and close buttons on the right. You can change the text that appears on the Title Bar, and adjust the size of the ***TC*** window, with the WINDOW 392 command. You can also adjust the size of the ***TC*** window using standard window techniques, but see Resizing the Take Command Window 84 for information about how ***TC***'s display changes when you do so.

2. See Take Command Menus 75 for details about the **Menu Bar** and all of its menus.

3. The **Tool Bar** is used to execute internal or external commands, aliases, batch files, and applications with the click of a mouse. You can define up to 32 Tool Bar buttons; see Tool Bar Dialog 79 for instructions. You can show or hide the Tool Bar with a choice on the Options Menu 77, from Window Option Tab 153 of the Configuration Dialog 151, or with the ToolBarOn 147 directive in TCMD32.INI 91.

4. The **Command Window** accepts your input and displays ***TC***'s output. You can use the scroll bars or the **Up Arrow** and **Down Arrow** keys to view text that has scrolled through the window. You can also save the contents of the Command Window and scrollback buffer to a file, copy text from Command Window to the clipboard, and copy text from the clipboard or from the Command Window to the command line. See Highlighting and Copying Text 83 for information about saving and retrieving text in the Command Window and The Command Line 46 for complete details about using the Command Line.

5. Finally, the **Status Bar** at the bottom of the ***TC*** window displays information about your system:

   - The date and time, based on the Windows clock.
   - The percentage of CPU usage.
   - The percentage "memory load" as reported by Windows.
   - The state of the Caps Lock key on the keyboard.
   - The state of the Num Lock key on the keyboard.
   - A slider control to adjust the transparency level of the ***TC*** window.

You can show or hide the Status Bar with a choice on the Options Menu 77, with the Window Option Tab 153 of the Configuration Dialog 151, or with the StatusBarOn 145 directive in TCMD32.INI 91.

If you find the "I-Beam" cursor in the ***TC*** window difficult to see, disable it from the **Startup** page of the configuration dialog, or set the IBeamCursor 123=No directive in the [TakeCommand] section of TCMD32.INI 91 to force the use of an arrow cursor in all parts of the window.

4.1.1 Take Command Menus

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

The items on the menu bar allow you to select a variety of ***TC*** features:

4.1.1.1 File Menu

The File menu allows you to save or print the screen buffer, or exit ***TC***.

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

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

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

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

Log Off
Exits ***TC*** and logs off the current user.

Shutdown
Exits ***TC*** and shuts down Windows.

Restart *Windows*
Exits ***TC***, shuts down Windows and then reboots the system.

Exit

Ends the current ***TC*** process.

4.1.1.2 Edit Menu

The Edit menu allows you to copy text between the ***TC*** window and the Windows clipboard. You can also access the clipboard with redirection 36 to or from the CLIP: device, or with the @CLIP 442 variable function.

To use the Cut, Copy, or Delete commands, you must first select a block of text with the mouse, the keyboard, or with the Select All command, below. If you hold down the right mouse button while you select a block of text, that block will be copied to the clipboard automatically when you release the button.

For more information on copying text see Highlighting and Copying Text 83.

Cut
Removes text from the ***TC*** command line and moves it to the clipboard.

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

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

Delete
Removes text from the ***TC*** command line.

Copy + Paste
Copies the selected text from the ***TC*** scrollback buffer directly to the command line.

Copy + Paste + Run
Copies the selected text from the ***TC*** scrollback buffer directly to the command line and executes the resulting command line.

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

Select All
Marks the entire contents of the ***TC*** scrollback buffer as selected text.

4.1.1.3 Apps Menu

Run
Displays the run dialog box 79 from which you can run an application or batch file. ***TC*** remembers the commands you have run from this dialog in the current session. To select from this list click on the drop-down arrow to the right of the "Command Line" field, or press the down-arrow.

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

View Console
Toggles the ***TC*** console window 86 to be visible or hidden. ***TC*** creates the console window to run

character-mode applications. Many such applications can be run under ***TC***'s Caveman 86 support, and will then display their output directly in the ***TC*** window. You can also control whether the console window remains hidden after an application finishes; see the **Console Hide** option on the Windows tab 153 of the configuration dialogs 151, or the HideConsole 120 directive in the .INI file 91.

4.1.1.4 Options Menu

Configure Take Command
Opens a dialog 151 which you can use to change the configuration of ***TC***.

Configure Tool Bar
Opens a dialog in which you can define up to 32 buttons for the ***TC*** tool bar 82.

Command Log
Enables or disables command logging using the default log file (*TCLOG*) or the file you have chosen with the LOG 303 command or the LogName 131 directive in the .INI file 91.

History Log
Enables or disables command history logging using the default log file (*TCHLOG*) or the file you have chosen with the LOG /H 303 command or the HistLogName 121 directive in the .INI file 91.

Session Log
Enables or disables session (all output) to the session log file (same name as the **Command Log** file but with a **.all** extension appended).

Show Tool Bar
Enables or disables the ***TC*** tool bar 82, which appears near the top of the ***TC*** window. The tool bar will not appear until you have defined at least one item for it with the **Configure Tool Bar** command, above.

Show Status Bar
Select this item to enable or disable the ***TC*** status bar 82, which appears near the bottom of the ***TC*** window.

4.1.1.5 Utilities Menu

Find Files/Text
Opens the Find Files Dialog Box 80 which lets you search for files or text interactively (see FFIND 262 to search from the command line).

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

Descriptions
Opens the Descriptions Dialog 81 edit window in which you can view and change the descriptions of files in any directory available on your system. See DESCRIBE 230 for details on file descriptions.

Aliases
Opens the Aliases 81 edit window in which you can view and change the list of current aliases 187. You can also use this window to import aliases from a file or to save the aliases to a file.

Environment
Opens the Environment 81 edit window in which you can view and change the current environment 395. You can also use this window to import environment variables from a file or to save the environment variables to a file.

Functions
Opens the User Functions 82 edit window in which you can view and change the list of current user-defined functions 275. You can also use this window to import user functions from a file or to save the user functions to a file.

Editor
Starts the Windows Notepad editor or any other editor you have specified with the Editor 116 directive in TCMD32.INI 91 or on the Misc tab 157 of the configuration dialogs 151.

4.1.1.6 Help Menu

See also: the Help File 506 topic.

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

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

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

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

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

About
Displays ***TC*** version, copyright, and license information.

4.1.2 Take Command Dialogs

The ***TC*** menus lead to several dialog boxes. Each is listed here for quick reference, though in general you will find it easier to learn about each one from the context in which it is used (for example, the information referenced below on the tool bar dialog will be more useful after you have read the section on the tool bar 82).

TC uses standard Windows dialogs for tasks like printing, selecting a font, or browsing files and

directories. These dialogs are provided by Windows, not ***TC***, and are common to many different Windows programs; they are not documented within this help system.

The reference in parentheses after certain dialogs listed below shows the name of the menu 75 you can use to access that dialog.

Run Program Dialog 79 (Apps 76)
Properties Dialog 151 (Options 77)
Startup Tab 152
Windows Tab 153
Editing Tab 154
Colors Tab 154
History Tab 155
Syntax Tab 156
Misc Tab 157
Internet Tab 156
Email Tab 156
Caveman Tab 158
Tool Bar Dialog 79 (Options 77)
Find Files/Text Dialog 80 (Utilities 77)
Edit Descriptions Dialog 81 (Utilities 77)
Aliases 81 (Utilities 77)
Environment Variables 81 (Utilities 77)
User Functions 82 (Utilities 77)

4.1.2.1 Run Program Dialog

The Run Program dialog, started from the Apps menu 76, allows you to run a program by typing its name or browsing the disk.

In the Command Line edit box, you can enter the name of any executable program plus command line parameters. If you click on the arrow to the right of the edit box, the dialog displays a list of previous commands you have entered during the current ***TC*** session.

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

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

4.1.2.2 Tool Bar Dialog

This dialog, available from the ***TC*** Options menu 77, allows you to define or modify buttons on the tool bar 82.

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

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

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

bar.

In the **Command** field, enter the command to be executed when you select the button on the tool bar. You can enter multiple commands 65 in the Command field by separating them with the command separator character **&**. You can use the **Browse** button to find a path and filename to be entered at the beginning of the Command field.

If the command line begins with an at-sign @ it will not be added to the command history 49 when the tool bar button is clicked. Otherwise, all commands executed from the tool bar are stored in the history.

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

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

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

If you exit by choosing the **OK** button, any changes you have made will be saved in TCMD32.INI 91, and reloaded automatically the next time you start ***TC***. If you use the **Cancel** button, your changes will be discarded.

If you want to rearrange the order of the buttons on the tool bar, use a text editor (*e.g.* Windows' Notepad) to edit the [Buttons] section of TCMD32.INI 91. Simply rearrange the lines into the order you wish, and renumber the buttons accordingly.

Alternatively, the tool bar can be configured with the TCTOOLBAR 374 command.

4.1.2.3 Find Files/Text Dialog

The Find Files/Text dialog box, available from the Utilities menu 77, gives you the same features as the FFIND 262 command, in dialog form.

Enter the file name or names you wish search in the **Files** field. You can use wildcards 19 and include lists 30 as part of the file name. To select files from previous searches in the same ***TC*** session, click on the down arrow beside the Files field, or press the up or down arrow while the input cursor is in the Files field. You can also use the **Browse** button to find files to include in the search.

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

If you use wildcards to specify the files to search, you can narrow the search with the **Exclude** field by specifying files that you want to exclude from the search. Like the **Files** field, the **Exclude** field can contain wildcards 19 and include lists 30. For example, if you want to search all files with an extension beginning with ***I*** except for ***.ICO*** and .INI file 91s, you could enter ****.i**** in the **Files** field

and then **.ico;*.ini* in the **Exclude** field.

Enter the text (or hexadecimal values) you are searching for in the **Text** field. You can use extended wildcards 19 in the search string to increase the flexibility of the search. Use back-quotes ` around the text if you want to search for characters which would otherwise be interpreted as wildcards. For example, to search for an **A**, followed by some number of other characters, followed by a **B**, enter the **A*B** as your search string. To search for the literal string **A*B** (**A**, followed by an asterisk, followed by **B**), enter **`A*B`** as your search string search string (the closing back-quote is optional).

The **Match Case** box, when it is selected, makes the search case-sensitive. This option is ignored if **Hex Search** is selected. The **Hex Search** option signals that you are searching for hexadecimal values, not ASCII characters. See the FFIND 262 command for more details.

If you enable **All Lines**, every matching line from every file will be displayed; otherwise only the first matching line from each file will be displayed. Unless you enable the **Hidden Files** option, files with the hidden and system attributes 524 will not be included in the search.

The radio buttons in the **Search** area let you specify where you want ***TC*** to look for files. If you select **Dir Only** or **Dir & Below**, the search will begin in the current default directory, shown above the Files box. If you select **Entire Disk**, ***TC*** will use the drives that you specified in the **Disks** field. If you select **All Hard Disks**, ***TC*** will search all the hard disk drives it finds on your system.

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

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

If you select one of the matching files in the list (by double-clicking on it, or selecting it with the cursor and pressing Enter), ***TC*** will display another dialog with complete directory information about the file. From that dialog you can **Run** the file (if it is an executable file, a batch file, or has an executable extension 33), display the file with the LIST 298 command, or **Edit** the file. If you choose **List**, the cursor will be placed on the first matching text within the file. When you exit from LIST 298 or the editor, the original list of matching files will still be available.

4.1.2.4 Edit Descriptions Dialog

This dialog is available from the Utilities menu 77. Most of it looks and works like a standard file browser. The pane at the bottom of the dialog lets you view, enter, or edit the description for any file. See DESCRIBE 230 for more information on file descriptions.

File descriptions can also be entered or changed with the DESCRIBE 230 command, and are visible when you use the DIR 233 and SELECT 345 commands.

4.1.2.5 Aliases Window

This dialog is available from the Batch Debugger 201, by using the ESET 258 /W command, or from the Utilities menu 77 (***TC*** only). You can use this window to view, edit, add, and delete aliases 187.

4.1.2.6 Environment Window

This dialog is available from the Batch Debugger 201, by using the ESET 258 /W command, or from the Utilities menu 77 (***TC*** only). You can use this window to view, edit, add, and delete environment variables 395.

4.1.2.7 Functions Window

This dialog is available from the Batch Debugger 201, by using the ESET 258 /W command, or from the Utilities menu 77 (***TC*** only). You can use this window to view, edit, add, and delete user-defined functions 275.

4.1.3 Tool Bar

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

To create buttons for the Tool Bar, select Configure Tool Bar from the Options 77 menu. This selection displays the tool bar dialog 79. You can also configure the Tool Bar with the TCTOOLBAR 374 command.

You can define up to 32 Tool Bar buttons.

To enable or disable the Tool Bar, use the ToolBarOn 147 directive in TCMD32.INI 91, the Tool Bar Enable setting on the Windows tab 153 in the configuration dialogs 151, or the Show Tool Bar command in the Options menu 77.

The configuration dialog and TCMD32.INI 91 settings are modified when you enable and disable the tool bar from the Options menu. This preserves the tool bar state when you close ***TC***, and restores it the next time you start a ***TC*** session.

4.1.4 Status Bar

The ***TC*** window has an optional Status Bar that shows information about your system. To enable or disable the Status Bar, use the StatusBarOn 145 directive in the TCMD32.INI 91 file, the Status Bar Enable setting on the Windows tab 153 in the configuration dialogs 151, or the **Show Statusbar** command in the Options menu 77.

The configuration dialog and TCMD32.INI 91 settings are modified when you enable and disable the status bar from the Options menu. This preserves the status bar state when you close ***TC***, and restores it the next time you start a ***TC*** session.

The status bar displays the following information:

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

4.1.5 Using the Scrollback Buffer

TC retains the text displayed on its screen in a "scrollback buffer". You can scroll through this buffer using the mouse and the vertical scroll bar at the right side of the ***TC*** window, just as you can in any Windows application. You can also use the **Up** and **Down** keys to scroll the display one line at a time from the keyboard, and the **PgUp** and **PgDn** keys to scroll one page at a time.

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

If you prefer to use the **Up/Down** and **PgUp** keys to access the command history (as in ***4NT***), see the SwapScrollKeys 145 directive in TCMD32.INI 91, or the corresponding option on the History tab 155 of the configuration dialogs 151. SwapScrollKeys 145 switches the keystroke mapping so that the **Up**, **Down**, and **PgUp** keys manipulate the command history, and **Ctrl-Up**, **Ctrl-Down**, **Ctrl-PgUp**, and **Ctrl-PgDn** are used to control the scrollback buffer. For more details see Scrolling and History Keystrokes 51.

You can set the size of the scrollback buffer on the Windows tab 153 of the configuration dialogs 151, or with the ScreenBufSize 141 directive in the TCMD32.INI 91 file.

To clear the entire scrollback buffer, use the CLS 215 /C command.

4.1.6 Highlighting & Copying Text

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

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

To copy text from the ***TC*** window to the clipboard, first use the mouse to highlight the text, then press **Ctrl-C**, or use the **COPY** command on the Edit menu 76.

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

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

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

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

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

If you prefer, you can configure ***TC*** to use the CUA keystrokes **Ctrl-Ins**, **Ctrl-Del**, and **Ctrl-Shift-Ins**

for Copy, Cut, and Paste on the Editing tab 154 of the configuration dialogs 151, or with the CUA 111 directive in the TCMD32.INI 91 file.

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

You can also use Windows' Drag and Drop 87 facility to paste a filename from another application onto the command line, and you can access the clipboard with redirection 36 to or from the CLIP: device, or with the @CLIP 442 variable function.

4.1.7 Resizing the Take Command Window

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

When the height of the command window changes, future commands simply use the new height as you see it on the screen. For example, if you reduce the window to three rows high and do a DIR /P 233 (display a directory of files and pause at the bottom of each visual "page"), DIR will display two lines of output, a prompt ("Press any key to continue ..."), and then pause. If you expand the window to 40 lines high and repeat the same command, DIR will display 39 lines, a prompt, and then pause.

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

The screen height normally starts at 25 lines; you can alter this default with the ScreenRows 142 directive in TCMD32.INI 91, or the **Height** setting on the Windows tab 153 of the configuration dialogs 151. The _ROWS 421 internal variable can be used to determine the current screen height.

The virtual screen width starts at 80 columns or the number of columns which fit into the startup ***TC*** window, whichever is larger. You can alter the default minimum width of 80 columns with the ScreenColumns 142 directive in the TCMD32.INI 91 file, or the **Width** setting on the Windows tab 153 of the configuration dialogs 151. The _COLUMNS 413 internal variable can be used to determine the current virtual screen width.

If you use keyboard commands or the mouse to expand the ***TC*** window beyond its previous virtual width, the virtual width is automatically increased. This ensures that the internal buffer can hold lines which will fill the newly enlarged window. If you contract the window, the virtual width is not reduced because this might require removing output already on the screen or in the scrollback buffer.

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

When the font is changed, ***TC*** will recalculate the virtual screen width. The new virtual width will be the width set by the Screen Columns directive or the configuration dialogs, or the current width of the

window in the new font, whichever is larger.

4.2 Using a Windows Command Line

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

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

What's new:

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

- A built-in scrollback buffer 83 that lets you look back through the output from past commands.
- A standard Windows menu bar 75 for access to many commonly-used ***TC*** features.
- A status bar 82 showing the date, time, keyboard toggles, and the cpu and memory usage.
- A customizable tool bar 82 that gives you quick access to commands and applications.
- Windows dialogs (accessible from the Options 77 and Utilities 77 menus) for editing environment variables, aliases, user functions, file descriptions, and startup parameters.
- Dialog-based file and text search (see "Find Files/Text" on the Utilities menu 77). The FFIND 262 command gives you the same capabilities at the ***TC*** prompt.
- A unique technology called Caveman 86, which you can use to run most character-mode applications in the ***TC*** window (see Console Applications and the Console Window 86 for details).

What's Different:

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

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

There are also some differences between running under ***4NT*** (or ***CMD.EXE***) and running under ***TC***. The primary differences are related to different methods for starting character-mode programs; this topic is covered in detail under Console Applications and the Console Window 86.

In order to support the ***TC*** window scrollback buffer, some ***TC*** keystrokes are different from what you may be used to in our character-mode products. See Scrolling and History Keystrokes 51 for more details.

Some command line editing 47 defaults have also been changed to conform more closely to

Windows conventions. In ***TC*** the default editing mode is insert, not overstrike, and the default insert-mode cursor is a line, not a block. You can change these defaults with statements in TCMD32.INI 91 or via the Editing tab 154 of the configuration dialogs 151.

4.3 Console Applications & the Console Window

The console session connected to ***TC*** is created when ***TC*** starts. You can view this window at any time with the **Alt-V** key or the **View Console** selection on the **Apps** menu. You can use **Alt-V** to return to the ***TC*** window, but only when the application in the console window has completed, and the input "focus" has returned to ***TC***. You can control the width and height of the console window with the ConsoleColumns 110 and ConsoleRows 111 directives in the TCMD32.INI 91 file, or the corresponding options on the Windows tab 153 of the configuration dialogs 151.

When you start a DOS or Windows character-mode application from ***TC***, it is automatically run in this console session (to start a separate session for the application, use the START 364 command). The console window automatically becomes visible when the application starts, and is hidden when the application exits. To leave the console window visible after the application exits, set the HideConsole 120 directive to No in TCMD32.INI 91, or set or the corresponding option on the Windows tab 153 of the configuration dialogs.

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

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

To make it easier to use character-mode applications from within Windows, ***TC*** also includes a technology called "Caveman." Caveman allows DOS and Windows 32-bit character-mode programs to display output and accept input within the ***TC*** window, and eliminates the requirement that you switch between the ***TC*** window and the console window. See the section on Caveman 86 for details.

4.4 Caveman (technical information)

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

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

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

When Caveman is running in ***TC***, it continually scans the console window 86 and updates the ***TC*** window with any changes it finds. It also sends keystrokes from the ***TC*** window to the console window. Caveman makes a character mode program appear to run in the ***TC*** window even though it is, in reality, running in the console session window.

Not all character-mode programs work well with Caveman. DOS programs which switch into graphics

mode or an unusual text mode or which manipulate the video or keyboard hardware directly are likely to fail. Unfortunately, ***TC*** cannot determine automatically which method is best for any given application. Therefore, you must enable Caveman for the applications you wish to use it with. While many applications will work properly, some experimentation on your part may be required.

Caveman also allows you to control whether the application's output is displayed using ***TC***'s default screen colors or the colors set by the application.

When you install ***TC***, Caveman is disabled. To enable it, use the Caveman tab 158 from the Configure ***TC*** item on the Options menu. The same tab lets you list the applications you wish to run under Caveman, and select how colors are handled (see the Caveman dialog help 158 for more details on these options). You can also use filenames and wildcards to include or exclude anything from an individual application to all programs on your system.

We recommend that you start by enabling all applications (use a "*****" entry as described in the dialog documentation), then add exceptions for applications which do not work well with Caveman.

Once an application is listed in the dialog and enabled, it will run under Caveman, within the ***TC*** window, whenever you start it from the command line. However, if you start the application with the START 364 command, the program will start in its own window.

You can also use START 364's /CM switch to run an application under Caveman, even if it is not one of the applications listed implicitly or explicitly in the Caveman Applications Dialog.

4.5 Using Drag & Drop

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

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

TC is a drag and drop "client", which means it can accept files dragged in from other applications and paste their names onto the command line as described above. It is not a drag and drop "server", so you cannot drag filenames from the ***TC*** window into other applications. However you can copy filenames and other text from the ***TC*** Window to other applications using the clipboard; see Highlighting and Copying Text 83 for details.

4.6 DDE Support

TC can communicate with other Windows applications by using Dynamic Data Exchange or **DDE**. To use ***TC*** as a DDE client and send a message from ***TC*** to another application, see the DDEEXEC 224 command.

You can also use ***TC*** as a DDE server and send commands to it from another application. To do so, use an application or server name of **TCMD32**, and a topic name of **Execute**. The message can be any valid command that you could enter on the Command line: any alias, internal command, batch file, or external command. To send more than one line in a single DDE string, separate the lines with carriage return and line feed characters.

When using ***TC*** as a DDE Server you should start ***TC***, execute the desired command, and exit ***TC***, using the DDE commands of the "client" application. ***TC***'s DDE server feature is not intended to be used with a copy of ***TC*** running at the prompt; attempting to do so may result in unusual displays or

display of the prompt in an incorrect location.

For example, if you use Microsoft Word you could use the following Visual Basic for Applications (VBA) fragment to send a DIR 233 /W command to ***TC*** from within Word:

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

You could use the same approach to execute a batch file or any other ***TC*** command from within Word, and similar approaches are available in other applications which offer DDE support. Consult your application manual for complete details.

In most cases, when you use ***TC*** as a DDE server you will want to redirect 36 output from the commands you execute, because output on the ***TC*** window is not likely to be useful to the client program which invokes a command.

5 Configuration Options

4NT and ***TC*** offer a wide range of configuration options, allowing you to customize command processor operation for your needs and preferences. Most of these options can be set through the use of directives 91 stored in Initialization Files 91. The SETDOS 354 command is available to dynamically set those options not settable through directives, and also to dynamically modify the effects of a few of the directives.. The OPTION 317 command can invoke the Configuration Dialog 151, which is able to set the majority of directives, or to dynamically modify the value of a single directive at a time.

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

- With aliases and user-defined functions you can set default options for internal commands and create new commands (see Aliases 160 and the ALIAS 187 and FUNCTION 275 commands).
- With executable extensions 33 you can associate data files with the applications you use to open them.
- With the FILECOMPLETION 399 environment variable and the FileCompletion 118 directive, you can customize filename completion to match the command you are working with.
- With the COLORDIR 398 environment variable and the ColorDir 98 directive you can set the colors used by the DIR 233 command.
- With command line options 4 you can specify where the command processor looks for its startup files and how it operates for a specific instance.

5.1 Initialization (.INI) Files

Part of the power of the command processor is its flexibility, in allowing you to alter its configuration to match your style of computing. Most of the command processor configuration is controlled through a file of initialization information.

See Locating the .INI files 89 below to find out how the command processor locates the *.INI* file.

Note: The *.INI* file is also used by the command processor to store some operating characteristics, such as debugger controls and the default size and location of the command processor window. This take place transparently to the user.

Modifying the .INI File

You can create, add to, and modify the .INI file in two ways:

- with the configuration dialog 151, available via the OPTION 317 command, and in ***TC*** also via the **Configure Take Command** selection on the Options menu 77, or
- by editing the file with any text editor.

Most of the changes you make in the configuration dialog 151 or with the OPTION 317 command take effect immediately. A few (*e.g.*, those associated with the startup screen size) only take effect when you start a new command processor session. See the online help for each individual dialog page if you are not sure when a change will take effect.

The dialogs handle most *.INI* file directives. The Advanced directives 98, the Key Mapping directives 102, and a few other individual directives (identified in the help topics for those directives) do not have corresponding fields in the configuration dialogs, and must be entered manually.

The command processor reads its *.INI* file (see Locating the .INI files 89) when it starts, and configures itself accordingly. The *.INI* file is not reread when you change it manually. For manual changes to take effect, you must restart the command processor. If you edit the *.INI* file manually, make sure you save the file in ASCII format.

Each item that you can include in the *.INI* file has a default value. You only need to include entries in the file for settings that you want to change from their default values.

Using the .INI File

Some settings in the *.INI* file are initialized when you install the command processor. Other (such as window size and position) are modified as you use the command processor, so you will have an *.INI* file even if you didn't create one yourself. You should not delete this file.

Locating the .INI File

1) When starting a primary shell 535

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

- First, the default *.INI* file name in the table below is used, and

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

  the search starts in the directory where the command processor program file is stored. This is the default location for the *.INI* file. The command processor determines this directory automatically.

- If the *.INI* file is not found in the directory where the program file is stored, the command processor searches the root directory 570 of the boot drive 557.

- If there is an ***@d:\path\inifile*** option on the startup command line, the command processor will use the path and file name specified there, and will overwrite any values set by the default *.INI* file (if any).

If no *.INI* file is found, all directives are set to their default values. If the operations performed in the command processor's window require it, a new *.INI* file will be created, using the default location and name, as explained above.

2) When starting a secondary shell

The command processor retrieves the primary shell's .INI file data, processes the [Secondary] section of the original *.INI* file if necessary, and then processes any ***@d:\path\inifile*** option on the secondary shell command line. You can override this behavior with the NextINIFile directive.

See Command Line Options for more details about the startup command line.

.INI File Sections

The *.INI* file has three common sections. The first is a global section, named after your command processor (**[4NT]** or **[TakeCommand]**); the **[Primary]** section; and the **[Secondary]** section. Each section is identified by the section name in square brackets on a line by itself.

Directives in the global section are effective in all shells. In most cases, this is the only section you will need. Any changes you make to the .INI file with the OPTION command are stored in the global section.

The **[Primary]** and **[Secondary]** sections include directives that are used only in primary and secondary shells, respectively. You don't need to set up these sections unless you want different directives for primary and secondary shells.

Directives in the **[Primary]** section are used for the first or primary shell. The values are passed automatically to all secondary shells, unless overridden by a directive with the same name in the **[Secondary]** section.

Directives in the **[Secondary]** section are used in secondary shells only, and override any corresponding primary shell settings. For example, these lines in the *.INI* file:

```
[Primary]
ScreenRows = 25
[Secondary]
ScreenRows = 50
```

mean to assume that you have 25 rows on the screen in the primary shell and 50 lines in all secondary shells.

See Primary and Secondary Shells for an explanation of those terms

The command processors may also create and maintain some other sections, e.g., **[Debugger]**, **[EnvironmentVariables]**, **[Frame]**.

Sections whose name is not known to the command processor are ignored, permitting you to add blocks of comments, such as an update history.

Note: The only significance of sections is whether or not directives in a specific section are used or ignored. The rules about precedence or cumulative effect, described in the Directives topic, are applicable to sections used.

5.2 Directives

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

- Initialization Directives 101
- Configuration Directives 99
- Color Directives 98
- Key Mapping Directives 102
- Advanced Directives 98

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

Syntax for Directives

Most lines in the *.INI* file consist of a one-word **directive**, an equal sign **=**, and a **value**. For example, in the following line, the word **History** is the directive and **2048** is the value:

```
History = 2048
```

Any spaces before or after the equal sign are ignored.

Regardless of how long a string value is, for example the list for the ColorDir 398 directive, you must enter it all on one line. Strings cannot be continued to a second line.

Each line must be within the command line length limit 71.

The format of the **value** part of a directive line depends on the individual directive. It may be a numeric value, a single character, a choice (like **Yes** or **No**), a color setting, a key name, a path, a filename, or a text string. The value begins with the first non-blank character after the equal sign and ends at the end of the line or the beginning of a comment.

Blank lines are ignored in the *.INI* file and can be used to separate groups of directives.

You can place comments in the file by beginning a line with a semicolon **;**. You can also place comments at the end of any line except one containing a text string value. To do so, enter at least one space or tab after the value, a semicolon, and your comment, like this:

```
History = 2048          ;set history list size
```

If you try to place a comment at the end of a string value, the comment will become part of the string and will probably cause an error.

If you use the configuration dialogs 151 to modify the *.INI* file, comments on lines modified from within the dialogs will not be preserved when the new lines are saved. To be sure *.INI* file comments are preserved, put them on separate lines in the file.

When the command processor detects an error while processing the *.INI* file, it displays an error message and prompts you before processing the remainder of the file. This allows you to note any errors before the startup process continues. The directive in error will retain its previous or default value.

If you need to test different values for an *.INI* directive without repeatedly editing the *.INI* file, use the OPTION 317 command or see the INIQuery 124 directive.

If you want to include the text of one *.INI* file within another (for example, if you have a set of common directives used by several JP Software products), use the Include 123 directive.

The SETDOS 354 command can override several of the *.INI* file directives. For example, the cursor shape used by the command processor can be adjusted either with the CursorIns and CursorOver directives or the SETDOS 354 /S command. The correspondence between a SETDOS 354 option and a *.INI* directive is noted under both the individual help topic for that directive and under that option in the SETDOS 354 help topic.

Secondary shells 535 automatically inherit the configuration settings currently in effect in the previous shell. If values have been changed by SETDOS 354 or OPTION 317 since the primary shell started, the current values will be passed to the secondary shell. If the previous shell's *.INI* file had a [Secondary] section, it will then be read and processed. If not, the previous shell's settings will remain in effect.

For example, you might set BatchEcho 107 to **Yes** in the *.INI* file to enable batch file echoing. If you turn off batch file echoing in the primary shell, then any secondary shells will inherit the current setting, rather than the original value from the *.INI* file; *i.e.*, batch files in the secondary shell will default to no echo.

If you want to force secondary shells to start with a specific value for a particular directive, regardless of any changes made in a previous shell, repeat the directive in the **[Secondary]** section of the *.INI* file.

Types of Directives

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

- **Name = *nnnn* (1234)**: This directive takes a numeric value which replaces the "nnnn." The default value is shown in parentheses or listed below the directive's description.

- **Name = *c* (X)**: This directive accepts a single character as its value. The default character is shown in parentheses. You must type in the actual character; you cannot use a key name.

- **Name = *CHOICE1* | *Choice2* | *...*** : This directive must be set to one of the vertical bar separated values listed between the braces. The default value is shown in all upper case letters in the directive description, but in your file any one of the choices can be entered, using any case. (Do not enter the vertical bar.) For example, if the choices were shown as **YES** | **No** then **YES** is the default.

- **Name = *Color***: This directive takes a color specification. See Colors and Color Names 552 for the format of color names.

- **Name = *Key*** : This directive takes a key specification. See Keys and Keynames 550 for the format of key names.

- **Name = *Path*** : This directive takes a path specification, without a filename. The value should include both a drive and path (*e.g.*, *C:\TCMD*) to avoid any possible ambiguities. A trailing backslash **** at the end of the path name is accepted but not required. Any default path is described in the text.

- **Name = *File*** : This directive takes a filename. We recommend that you use a full filename including the drive letter and path to avoid any possible ambiguities. Any default filename is described in the text.

- **Name = *String*** : This directive takes a string in the format shown. The text describes the default value and any additional requirements for formatting the string correctly. No comments are allowed.

- **Name** : This directive accepts NO parameters and the = is unnecessary (e.g. ClearKeymap 109).

Evaluation of Directives

The command processor contains a fixed-length area for storing strings entered in the *.INI* file, including file names, paths, and other strings. This area is large and is unlikely to overflow; if it does, you will receive an error message. If this occurs, reduce the complexity of your *.INI* file or contact our technical support department 503 for assistance.

The directives are evaluated sequentially from top to bottom within each section processed. When an initialization 101, configuration 99, or color 98 directive is processed more than once during startup, the each value replaces any previous one(s).

Most key mapping 102 and advanced 98 directives are cumulative and may appear several times when several concurrent values are desired, such as when assigning several different keystrokes to the same function.

5.2.1 Directives by Name

4StartPath 105	(***4NT***) Set location of 4START 8 and 4EXIT 8

AddFile 105	Keeps filename completion entry and adds another
AliasExpand 105	Expands aliases on command line
AmPm 105	Time display format
ANSI 105	Enables ANSI X3.64 support
AppendToDir 106	Appends trailing "\" to directory names in filename completion
AutoCancel 106	If yes, ^C will cancel batch files without a prompt
AutoRun 106	Run programs in AutoRun registry key

Backspace 106	Deletes the character to the left of the cursor
BatchEcho 107	Default echo state for batch files
BeepFreq 107	Default beep frequency
BeepLength 107	Default beep length
BeginLine 107	Moves the cursor to the start of the line
BGColorRGB 108	(***TC***) Set the default background color as an RGB value

CDDWinColors 108	Colors for directory search window.
CDDWinHeight 108	Initial height of the extended directory search window
CDDWinLeft 108	Initial position of the left border of the extended directory search window
CDDWinTop 108	Initial position of the top border of the extended directory search window
CDDWinWidth 109	Initial width of the extended directory search window
ClearKeyMap 109	Clear default key mappings
ColorDir 109	Directory colors

CMDExtensions 109	Enable or disable CMD.EXE-compatible command extensions
CommandEscape 110	Allows direct entry of a keystroke
CommandSep 110	Multiple command separator character
CompleteHidden 110	Enable / disable return of hidden files from filename completion
CompletePaths 110	Search PATH during filename completion
ConsoleColumns 110	(***TC***) Sets the width of the console mode screen buffer
ConsoleRows 111	(***TC***) Sets the height of the console mode screen buffer
Copy 111	Copies highlighted text to the keyboard
CopyPrompt 111	Enable or disable confirmation prompt for COPY 216 and MOVE 308
CUA 111	(***TC***) Key set used for cut, copy, and paste
CursorIns 112	Cursor width in insert mode
CursorOver 112	Cursor width in overstrike mode

Debug 112	Set debugging options
DebuggerTransparency 113	Transparency of the debugger window
DecimalChar 113	Select decimal separator for @EVAL, etc.
Del 113	Deletes the character at the cursor
DelayedExpansion 113	Enable or disable CMD.EXE-style delayed expansion
DelGlobalQuery 113	Enable or disable confirmation prompt with DEL /Q
DelHistory 114	Deletes a history list entry
DelToBeginning 114	Deletes from the cursor to the start of the line
DelToEnd 114	Deletes from the cursor to the end of the line
DelWordLeft 114	Deletes the word to the left of the cursor
DelWordRight 114	Deletes the word to the right of the cursor
DescriptionMax 115	Maximum length of file descriptions
DescriptionName 115	Name of file to hold file descriptions
Descriptions 115	Enable / disable description processing
DirHistory 115	Set size of Directory History
DirWinOpen 115	Opens the directory history window
Down 116	Moves the cursor or scrolls the display down
DuplicateBugs 116	Duplicate well-known CMD.EXE bugs

EditMode 116	Editing mode (insert / overstrike)
Editor 116	(***TC***) Program to run for "Editor" menu choice
EndHistory 116	Displays the last entry in the history list
EndLine 117	Moves the cursor to the end of the line
EraseLine 117	Deletes the entire line
EscapeChar 117	Select escape character.
EvalMax 117	Maximum precision displayed by @EVAL
EvalMin 117	Minimum precision displayed by @EVAL
ExecLine 118	Executes or accepts a line
ExecWait 118	Forces the command processor to wait for external programs to complete
ExitFile 118	Enable or disable 4EXIT / TCEXIT

FGColorRGB 118	(***TC***) Set the default foreground color as an RGB value
FileCompletion 118	Select files selected for file completion
FirewallHost 119	Firewall server name
FirewallPassword 119	Password for firewall authentication
FirewallType 119	Type of firewall
FirewallUser 119	Username for firewall authentication

FTPCFG 119	Specifies the location of the file containing the ftp user names and passwords
FTPTimeout 119	Timeout (inactivity) period for FTP / FTPS operations
FuzzyCD 120	Enables or disables extended directory searches

Help 120	Invokes this help system
HelpWord 120	Invokes help for the word at the cursor
HideConsole 120	(***TC***) Hide the console window after running a character-mode application
HistCopy 121	Copy recalled commands to end of history
HistDups 121	Controls entry/retention of duplicate history entries
HistFile 121	Persistent history file name
HistLogName 121	History log file name
HistLogOn 121	Turns on history logging when the command processor starts
HistMin 122	Minimum command length to save
HistMove 122	Move recalled commands to end of history
History 122	Set size of Command History
HistWinOpen 122	Opens the command history window
HistWrap 123	Behavior of the command history list
HTTPTimeout 123	Timeout (inactivity) period for HTTP / HTTPS operations

IBeamCursor 123	(***TC***) Set text cursor style
InactiveTransparency 123	(***TC***) Set the transparency level when out of focus
Include 123	Include text from another file in the current .INI file
INIQuery 124	Query for each line in the .INI file
InputColors 124	Input colors
Ins 124	Toggles insert / overstrike mode

JabberPassword 125	User password for Jabber server
JabberServer 125	Jabber server name
JabberUser 125	Jabber user name

LastHistory 125	Returns last history entry
Left 125	Moves the cursor or scrolls the display left
LFNToggle 125	Toggles between long and short filenames
LineToEnd 126	Copies a line to the end of the history, then executes it.
ListBack 126	Returns to the previous file
ListboxBarColors 126	(***4NT***) Colors of popup list boxes.
ListContinue 127	Go to the next file
ListColors 126	Colors used in the LIST 298 display
ListExit 127	Exits the current file
ListFind 127	Prompts and searches for a string
ListFindRegex 127	Prompt and search for a regular expression
ListFindRegexReverse 127	Prompt and search backwards for a regular expression
ListFindReverse 127	Prompts and searches backwards
ListHex 128	Toggles between hexadecimal and character display modes
ListHighBit 128	Toggles LIST's "strip high bit" option
ListInfo 128	Displays information about the current file
ListInverseColors 128	(***4NT***) Set the color to use in LIST for search matches
ListNext 128	Finds the next matching string
ListOpen 129	Displays the "open file" dialog

ListPrevious 129	Finds the previous matching string
ListPrint 129	Prints the file on the default printer
ListRowStart 129	Starting row number for LIST and FFIND
ListStatBarColors 130	(***4NT***) Color for LIST 298 status bar
ListUnicode 130	Toggles Unicode display mode
ListWrap 130	Toggles LIST's wrap option
LocalAliases 130	Local vs. global aliases
LocalDirHistory 130	Local vs. global directory history
LocalFunctions 131	Local vs. global functions
LocalHistory 131	Local vs. global history
LogAll 131	(***TC***) Log all output
LogErrors 131	Send error messages to the log file
LogName 131	Name the log file
LogOn 131	Turns on command logging when the command processor starts

MailAddress 132	Email address of user
MailPassword 132	Password for SMTP authentication
MailPort 132	SMTP port number
MailServer 132	Your SMTP server name
MailUser 132	User name for SMTP authentication

NextFile 133	Gets the next matching filename
NextHistory 133	Recalls the next command from the history
NextINIFile 133	.INI file for all secondary shells
NoClobber 133	Overwrite protection for output redirection
NormalEditKey 134	Deassigns a command line editing key
NormalKey 134	Deassigns a key
NormalListKey 134	Deassigns a LIST 298 window key
NormalPopupKey 134	Deassigns a popup window key
NTFSDescriptions 134	Store descriptions in NTFS SummaryInformation stream

ParameterChar 135	Alias / batch file parameter character
PassiveFTP 135	Enable or disable passive mode for FTP calls
Paste 135	Pastes line from clipboard
PathExt 135	Enable or disable the PATHEXT variable
PauseOnError 136	(***4NT***) Force pause after displaying error message
Perl 136	Enable or disable internal Perl support
PopFile 136	Opens the filename completion window
PopupWinBegin 136	Moves to the first line of the popup window
PopupWinColors 136	(***TC***) Colors for most popup windows
PopupWinDel 137	Deletes a line from within the popup window
PopupWinEdit 137	Moves a line from the popup window to the prompt
PopupWinEnd 137	Moves to the last line of the popup window
PopupWinExec 137	Selects the current item and closes the popup window
PopupWinHeight 137	Initial height of popup windows
PopupWinLeft 138	Initial position of left border of popup windows
PopupWinTop 138	Initial position of top border of popup windows
PopupWinWidth 138	Initial width of popup windows
PrevFile 138	Gets the previous matching filename
PrevHistory 139	Recalls the previous command from the history
Proxy 139	Sets name of HTTP Proxy server

ProxyPassword 139	HTTP proxy password for Basic authentication
ProxyPort 139	Sets port number of HTTP Proxy server
ProxyUser 139	HTTP proxy user name for Basic authentication

RecycleBin 139	Sets whether deleted files go to the Recycle Bin
RegularExpressions 140	Sets the regular expression syntax type
RepeatFile 140	Repeats previous match during filename completion
REXX 140	Enable or disable internal REXX support
Right 140	Moves the cursor or scrolls the display right
RLocalHost 140	Set name of local host for REXEC and RSHELL
RLocalUser 141	Set name of the user on the local machine for RSHELL
RLocalPort 141	Set the port to bind to for REXEC and RSHELL
Ruby 141	Enable or disable the internal Ruby support

SaveDirCase 141	Control preservation of original upper/lower case name when changing directories.
SaveHistory 141	Saves the command line without executing it
ScreenBufSize 141	(***TC***) Size of screen buffer
ScreenColumns 142	(***TC***) Virtual screen width
ScreenRows 142	(***TC***) Virtual screen height
ScrollDown 142	(***TC***) Scroll the buffer down one line.
ScrollPgDn 142	(***TC***) Scroll the buffer down one page.
ScrollPgUp 142	(***TC***) Scroll the buffer up one page.
ScrollUp 143	(***TC***) Scroll the buffer up one line.
SelectColors 143	Colors used in the SELECT 345 display
SelectStatBarColors 143	(***4NT***) Color used by SELECT 345 status bar
ServerCompletion 143	Sets server name completion options
SettingChange 143	Monitor WM_SETTINGCHANGE message
SHChangeNotify 143	Notify the system shell when changing files or directories
SSLPort 144	Sets the port number for FTP SSL service
SSLProvider 144	Sets the name of the security provider
SSLStartMode 144	Specifies how to start SSL negotiation
StartupFile 144	Enable or disable 4START / TCSTART execution
StatusBarOn 145	(***TC***) Set status bar mode at startup
StatusBarText 145	(***TC***) Point size of status bar text
StdColors 145	Standard display colors
SwapScrollKeys 145	(***TC***) Switch to 4NT-style history and scrolling keys
SwitchChar 146	Set the default switch character

TabStops 146	Sets the tab positions for output.
TCStartPath 146	(***TC***) Path for TCSTART 8 and TCEXIT 8
TFTPTimeout 146	Timeout (inactivity) period for TFTP operations
ThousandsChar 147	Thousands separator for @EVAL 451, etc.
TimeServer 147	URL of time server
ToolBarOn 147	(***TC***) Set tool bar mode at startup
ToolBarText 147	(***TC***) Point size of tool bar text
Transparency 148	(***TC***) Default transparency for Take Command window
TreePath 148	Path for directory database JPSTREE.IDX

UnicodeOutput 148	Use Unicode for output redirection
UnixPaths 148	Enable or disable forward slash in command paths

Up 149	Moves the cursor or scrolls the display up
UpdateTitle 149	Prevents or allows updating of the command processor window title

VariableExpand 149	Expand variables at the command line

Win32SFNSearch 149	Search for short and long file names
WindowHeight 149	Initial height of the command processor window
WindowState 149	Initial state for the command processor window
WindowWidth 150	Initial width of the command processor window
WindowX 150	Initial position of the left border of the command processor window
WindowY 150	Initial position of the top border of the command processor window
WordLeft 150	Moves the cursor left one word
WordRight 150	Moves the cursor right one word
Wow64FsRedirection 151	Enable or disable Windows 64 remapping of windows\system32

ZoneID 151	Enable or disable ZoneID security

5.2.2 Directives by Category

- Advanced Directives 98
- Color Directives 98
- Configuration Directives 99
- General Input Keys 103
- Initialization Directives 101
- Key Mapping Directives 102

5.2.2.1 Advanced Directives

These directives are generally used for unusual circumstances, or for diagnosing problems. Most often they are not needed in normal use. They cannot be entered via the configuration dialogs 151; you must enter them manually (see the .INI file 91 for details).

AutoRun 106	Run programs in AutoRun registry keys
ClearKeyMap 109	Clear default key mappings
Debug 112	Set debugging options
Include 123	Include text from another file in the current .INI file 91
NextINIFile 133	.INI file 91 for all secondary shells
SaveDirCase 141	Control preservation of original upper/lower case name when changing directories.

5.2.2.2 Color Directives

These directives control the colors that the command processor uses for its displays. For complete details on color names and numbers, see Colors and Color Names 552. The color directives are:

BGColorRGB 108	(***TC***) Set the default background color as an RGB value
CDDWinColors 108	Colors for directory search window.
ColorDir 109	Directory colors
FGColorRGB 118	(***TC***) Set the default foreground color as an RGB value

InputColors 124	Input colors
ListboxBarColors 126	(***4NT***) Colors of popup list boxes.
ListColors 126	Colors used in the LIST 298 display
ListInverseColors 128	(***4NT***) Set the color to use in LIST for search matches
ListStatBarColors 130	(***4NT***) Color for LIST 298 status bar
PopupWinColors 136	(***TC***) Colors for most popup windows
SelectColors 143	Colors used in the SELECT 345 display
SelectStatBarColors 143	(***4NT***) Color used by SELECT 345 status bar
StdColors 145	Standard display colors

5.2.2.3 Configuration Directives

These directives control the way that the command processor operates. Some can be changed with the SETDOS 354 command while the command processor is running. Any corresponding SETDOS 354 command is listed in the description of each directive. The configuration directives are:

AmPm 105	Time display format
ANSI 105	Enables ANSI X3.64 support
AppendToDir 106	Appends trailing "\" to directory names in filename completion
AutoCancel 106	If yes, ^C will cancel batch files without a prompt
BatchEcho 107	Default echo state for batch files
BeepFreq 107	Default beep frequency
BeepLength 107	Default beep length
CDDWinHeight 108	Initial height of the extended directory search window
CDDWinLeft 108	Initial position of the left border of the extended directory search window
CDDWinTop 108	Initial position of the top border of the extended directory search window
CDDWinWidth 109	Initial width of the extended directory search window
CommandSep 110	Multiple command separator character
CompleteHidden 110	Enable / disable return of hidden files from filename completion
CompletePaths 110	Search for filename completion in the PATH
CopyPrompt 111	Enable or disable confirmation prompt for COPY 216 and MOVE 308
CUA 111	(***TC***) Key set used for cut, copy, and paste
CursorIns 112	Cursor width in insert mode
CursorOver 112	Cursor width in overstrike mode
DebuggerTransparency 113	Set the transparency of the debugger window
DelayedExpansion 113	Enable / disable CMD.EXE-style delayed expansion
DelGlobalQuery 113	Enable or disable confirmation prompt with DEL /Q
DecimalChar 113	Select decimal separator for @EVAL, etc.
DescriptionMax 115	Maximum length of file descriptions
DescriptionName 115	Name of file to hold file descriptions
Descriptions 115	Enable / disable description processing
EditMode 116	Editing mode (insert / overstrike)
Editor 116	(***TC***) Program to run for "Editor" menu choice
EscapeChar 117	Select escape character.
EvalMax 117	Maximum precision displayed by @EVAL
EvalMin 117	Minimum precision displayed by @EVAL
ExecWait 118	Forces the command processor to wait for external programs to complete
FileCompletion 118	Select files selected for file completion
FirewallHost 119	Firewall server name
FirewallPassword 119	Password for firewall authentication
FirewallType 119	Type of firewall

FirewallUser 119	Username for firewall authentication
FTPCFG 119	Specifies the location of the file containing the ftp user names and passwords
FTPTimeout 119	Timeout (inactivity) period for FTP / FTPS operations
FuzzyCD 120	Enables or disables extended directory searches
HistCopy 121	Copy recalled commands to end of history
HistDups 121	Controls entry/retention of duplicate history entries
HistLogName 121	History log file name
HistLogOn 121	Turns on history logging when the command processor starts
HistMin 122	Minimum command length to save
HistMove 122	Move recalled commands to end of history
HistWrap 123	Behavior of the command history list
HTTPTimeout 123	Timeout (inactivity) period for HTTP / HTTPS operations
InactiveTransparency 123	(***TC***) Set the transparency level when out of focus
JabberPassword 125	User password for Jabber server
JabberServer 125	Jabber server name
JabberUser 125	User name for Jabber server
ListRowStart 129	Starting row number for LIST and FFIND
LogAll 131	(***TC***) Log all output
LogErrors 131	Send error messages to the log file
LogName 131	Name the log file
LogOn 131	Turns on command logging when the command processor starts
MailAddress 132	Email address of user
MailPassword 132	Password for SMTP authentication
MailPort 132	SMTP port number
MailServer 132	Your SMTP server name
MailUser 132	User name for SMTP authentication
NoClobber 133	Overwrite protection for output redirection
NTFSDescriptions 134	Use NTFS SummaryInformation stream for file comments
ParameterChar 135	Alias / batch file parameter character
PassiveFTP 135	Enable or disable passive mode for FTP calls
PathExt 135	Enable or disable the PATHEXT variable
Perl 136	Enable or disable internal Perl language support
PopupWinHeight 137	Initial height of popup windows
PopupWinLeft 138	Initial position of left border of popup windows
PopupWinTop 138	Initial position of top border of popup windows
PopupWinWidth 138	Initial width of popup windows
Proxy 139	Sets name of HTTP Proxy server
ProxyPassword 139	HTTP proxy password for Basic authentication
ProxyPort 139	Sets port number of HTTP Proxy server
ProxyUser 139	HTTP proxy user name for Basic authentication
RecycleBin 139	Sets whether deleted files go to the Recycle Bin
RegularExpressions 140	Set the regular expression syntax type
REXX 140	Enable or disable internal REXX language support
RLocalHost 140	Set name of local host for REXEC and RSHELL
RLocalPort 141	Set the port to bind to for REXEC and RSHELL
RLocalUser 141	Set name of the user on the local machine for RSHELL
Ruby 141	Enable or disable internal Ruby language support
ScreenColumns 142	(***TC***) Virtual screen width
ScreenRows 142	(***TC***) Virtual screen height

ServerCompletion 143	Sets server name completion options
SettingChange 143	Monitor the Windows WM_SETTINGCHANGE message
SHChangeNotify 143	Notify the system shell when changing files or directories
SSLPort 144	Sets the port number for FTP SSL service
SSLProvider 144	Sets the name of the security provider
SSLStartMode 144	Specifies how to start SSL negotiation
StatusBarText 145	(***TC***) Point size of status bar text
StatusBarOn 145	(***TC***) Set status bar mode at startup
SwapScrollKeys 145	(***TC***) Switch to 4NT-style history and scrolling keys
SwitchChar 146	Set the switch character
TabStops 146	Sets the tab positions for output.
TFTPTimeout 146	Timeout (inactivity) period for TFTP operations
ThousandsChar 147	Thousands separator for @EVAL 451, etc.
ToolBarOn 147	(***TC***) Set tool bar mode at startup
ToolBarText 147	(***TC***) Point size of tool bar text
Transparency 148	(***TC***) Transparency for Take Command window
UnixPaths 148	Enable or disable forward slash in command paths
UpdateTitle 149	Prevents or allows updating of the command processor window title
VariableExpand 149	Expand variables at the command line
Win32SFNSearch 149	Search for short and long file names
Wow64FsRedirection 151	Enable or disable Windows 64 remapping of windows\system32
ZoneID 151	Enable or disable NTFS ZoneID security

5.2.2.4 Initialization Directives

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

4StartPath 105	(***4NT***) Set location of 4START 8 and 4EXIT 8
CMDExtensions 109	Enable CMD.EXE-compatible extensions
ConsoleColumns 110	(***TC***) Sets the width of the console mode screen buffer
ConsoleRows 111	(***TC***) Sets the height of the console mode screen buffer
DirHistory 115	Set size of Directory History
DuplicateBugs 116	Duplicate well-known CMD.EXE bugs
ExitFile 118	Enable or disable 4START 8 or 4EXIT 8 (***4NT***) or TCSTART 8 and TCEXIT 8 (***TC***)
HideConsole 120	(***TC***) Hide the console window after running a character-mode application
HistFile 121	"Persistent" history filename
History 122	Set size of Command History
IBeamCursor 123	(***TC***) Set text cursor style
INIQuery 124	Query for each line in the .INI file
LocalAliases 130	Local vs. global aliases
LocalDirHistory 130	Local vs. global directory history
LocalFunctions 131	Local vs. global functions
LocalHistory 131	Local vs. global history
PauseOnError 136	(***4NT***) Force pause after displaying error message
ScreenBufSize 141	(***TC***) Size of screen buffer
StartupFile 144	Enable or disable 4START 8 and 4EXIT 8 (***4NT***) or TCSTART 8 and TCEXIT 8 (***TC***)

TCStartPath 146	(*TC*) Path for TCSTART 8 and TCEXIT 8
TimeServer 147	URL of time server
TreePath 148	Path for directory database JPSTREE.IDX
UnicodeOutput 148	Use Unicode for output redirection
WindowState 149	Initial state for the command processor window
WindowHeight 149	Initial height of the command processor window
WindowWidth 150	Initial width of the command processor window
WindowX 150	Initial position of the left border of the command processor window
WindowY 150	Initial position of the top border of the command processor window

5.2.2.5 Key Mapping Directives

These directives allow you to change the keys used for command line editing and other internal functions. They cannot be entered via the configuration dialogs 151; you must enter them manually (see the .INI file 91 topic for details).

They are divided into four types, depending on the context in which the keys are used. For a discussion and list of directives for each type see:

- General Input Keys 103
- Command Line Editing Keys 103
- TC Scrollback Buffer Keys 104
- Popup Window Keys 104
- LIST Keys 104

Using a key mapping directive allows you to assign a different or additional key to perform the function described. For example, to use function key **F3** to invoke the HELP 284 facility (normally invoked with **F1**):

```
Help = F3
```

Any directive can be used multiple times to assign multiple keys to the same function. For example:

```
ListFind = F            ;F does a find in LIST
ListFind = F4           ;F4 also does a find in LIST
```

Use some care when you reassign keystrokes. If you assign a default key to a different function, it will no longer be available for its original use. For example, if you assign **F1** to the AddFile 105 directive (a part of filename completion), the **F1** key will no longer invoke the help system, so you will probably want to assign a different key to Help.

See Keys and Key Names 550 before using the key mapping directives.

Key assignments are processed before looking for keystroke aliases. For example, if you assign Shift-F1 to HELP 284 and also assign Shift-F1 to a key alias, the key alias will be ignored.

Assigning a new keystroke for a function does not deassign the default keystroke for the same function. If you want to deassign one of the default keys, use the NormalKey 134, NormalEditKey 134, NormalPopupKey 134 or NormalListKey 134 directive. You must also deassign default keys before you can assign them to a different usage.

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

5.2.2.5.1 General Input Keys

These directives apply to all input. They are in effect whenever the command processor requests input from the keyboard, including during command line editing 47 and the DESCRIBE 230, ESET 258, INPUT 293, LIST 298, and SELECT 345 commands. The general input keys are:

Backspace 106	Deletes the character to the left of the cursor
BeginLine 107	Moves the cursor to the start of the line
Copy 111	Copies highlighted text to the keyboard
Del 113	Deletes the character at the cursor
DelToBeginning 114	Deletes from the cursor to the start of the line
DelToEnd 114	Deletes from the cursor to the end of the line
DelWordLeft 114	Deletes the word to the left of the cursor
DelWordRight 114	Deletes the word to the right of the cursor
Down 116	Moves the cursor or scrolls the display down
EndLine 117	Moves the cursor to the end of the line
EraseLine 117	Deletes the entire line
ExecLine 118	Executes or accepts a line
Ins 124	Toggles insert / overstrike mode
Left 125	Moves the cursor or scrolls the display left
NormalKey 134	Deassigns a key
Paste 135	Pastes line from clipboard
Right 140	Moves the cursor or scrolls the display right
Up 149	Moves the cursor or scrolls the display up
WordLeft 150	Moves the cursor left one word
WordRight 150	Moves the cursor right one word

5.2.2.5.2 Command Line Editing Keys

These directives apply only to command line editing 47. They are only effective at the prompt. The command line editing keys are:

AddFile 105	Keeps filename completion entry and adds another
AliasExpand 105	Expands aliases on the command line
CommandEscape 110	Allows direct entry of a keystroke
DelHistory 114	Deletes a history list entry
EndHistory 116	Displays the last entry in the history list
Help 120	Invokes this help system
HelpWord 120	Invokes help for the word at the cursor
LastHistory 125	Recall the last history entry
LFNToggle 125	Toggles between long and short filenames
LineToEnd 126	Copies a line to the end of the history, then executes it
NextFile 133	Gets the next matching filename
NextHistory 133	Recalls the next command from the history
NormalEditKey 134	Deassigns a command line editing key
PopFile 136	Opens the filename completion window
PrevFile 138	Gets the previous matching filename
PrevHistory 139	Recalls the previous command from the history
RepeatFile 140	Repeats previous match during filename completion
SaveHistory 141	Saves the command line without executing it
VariableExpand 149	Expand variables on the command line

5.2.2.5.3 TC Scrollback Buffer Keys

The following keys (available only in ***TC***), are also part of the command line editing group. They are used to manipulate the scrollback buffer 83 rather than to edit commands. For additional information see Scrolling 51 and History Keystrokes 51 and the SwapScrollKeys 145 directive.

ScrollUp 143	(TC) Scroll the buffer up one line
ScrollDown 142	(TC) Scroll the buffer down one line
ScrollPgUp 142	(TC) Scroll the buffer up one page
ScrollPgDn 142	(TC) Scroll the buffer down one page

See other Command Line Editing Keys 104.

5.2.2.5.4 LIST Keys

These directives are effective only inside the LIST 298 command.

ListBack 126	Return to the previous file
ListClipboard 126	Copy the current filename tot he clipboard
ListContinue 127	Continue to the next file
ListExit 127	Exits the current file
ListFind 127	Prompts and searches for a string
ListFindReverse 127	Prompts and searches backwards
ListFindRegex 127	Prompt and search for a regular expression
ListFindRegexReverse 127	Prompt and search backwards for a regular expression
ListHex 128	Toggles between hexadecimal and character display modes
ListHighBit 128	Toggles LIST's "strip high bit" option
ListInfo 128	Displays information about the current file
ListNext 128	Finds the next matching string
ListOpen 129	Displays the "open file" dialog
ListPrevious 129	Finds the previous matching string
ListPrint 129	Prints the file on the default printer
ListRefresh 129	Refresh the display
ListUnicode 130	Toggles Unicode display mode
ListWrap 130	Toggles LIST's wrap option
NormalListKey 134	Deassigns a LIST key

5.2.2.5.5 Popup Window Keys

The following directives apply to popup windows, including the command history window, the directory history window, the filename completion window, the extended directory search window, and the @SELECT 489 window.

DirWinOpen 115	Opens the directory history window
HistWinOpen 122	Opens the command history window
NormalPopupKey 134	Deassigns a popup window key
PopupWinBegin 136	Moves to the first line of the popup window
PopupWinDel 137	Deletes a line from within the popup window
PopupWinEdit 137	Moves a line from the popup window to the prompt
PopupWinEnd 137	Moves to the last line of the popup window
PopupWinExec 137	Selects the current item and closes the popup window

5.2.3 4StartPath (4NT)

4StartPath = *Path*

(4NT) Specifies the drive and directory where the 4START 8 and 4EXIT 8 programs (if any) are located.

See also: Automatic Startup and Termination Programs 8.

5.2.4 AddFile

AddFile = *Key*

Defaults:

TC	Ctrl-Shift-Tab
4NT	F10

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

See other Command Line Editing Keys 103.

5.2.5 AliasExpand

AliasExpand = *Key*

Default: Ctrl-F

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

See other Command Line Editing Keys 103.

5.2.6 AmPm

AmPm = yes | NO | auto

Yes	displays times in 12-hour format with a trailing "a" for AM or "p" for PM.
No	forces a display in 24-hour time format.
Auto	formats the time according to the country code set for your system.

Specifies the format of time displays in the output of the DATE 224, DIR 233, SELECT 345, TIME 378 and TIMER 379 commands, and in LOG 303 files. It has no effect on %_TIME 423, %@MAKETIME 479, the **$t** and **$T** options of PROMPT 329, or date and time ranges 22.

See other Configuration Directives 99.

5.2.7 ANSI

ANSI = yes | NO

Yes enables ANSI X3.64 string processing of the outputs of ***4NT*** / ***TC*** internal commands.

Note that ANSI X3.64 processing of the output of external applications is not available. Caveman output in ***TC*** will use the ANSI X3.64 default colors if the application is set to "Use default colors". See the ANSI X3.64 Commands Reference 550 for a list of the ANSI X3.64 commands supported by ***4NT*** and ***TC***. See also: the _ANSI 411 internal variable.

See other Configuration Directives 99.

5.2.8 AppendToDir

AppendToDir = yes | NO

Yes appends a trailing \ to directory names (or a trailing / to FTP URLs) when doing filename completion. If you have the UnixPaths 148 *.INI* directive set, the command processor will instead add a trailing forward slash to directory names. Regardless of the setting of this directive, a trailing backslash is always appended to a directory name at the beginning of the command line to enable automatic directory changes 17.

See other Configuration Directives 99.

5.2.9 AutoCancel

AutoCancel = yes | NO

Yes causes a ^C to cancel batch file processing without the usual prompt.

See other Configuration Directives 99.

5.2.10 AutoRun

AutoRun = YES | no

If **AutoRun**=YES ***4NT*** or ***Take Command*** start, they will look for and execute the following registry variables:

HKEY_LOCAL_MACHINE\Software\Microsoft\Command Processor\AutoRun

and / or

HKEY_CURRENT_USER\Software\Microsoft\Command Processor\AutoRun

See other Configuration Directives 99.

5.2.11 Backspace

Backspace = *Key*

Default: Backspace

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

See other General Input Keys 103.

5.2.12 BatchEcho

BatchEcho = YES | no

Controls the default batch echo mode. Yes enables echoing of all batch file commands unless ECHO 253 is explicitly set off in the batch file. No disables batch file echoing unless ECHO 253 is explicitly set on.

See also: SETDOS /V 354.

See other Configuration Directives 99.

5.2.13 BeepFreq

BeepFreq = *nnnn*

Default: 440

Specifies, in Hz, the default frequency for the BEEP 208 command. This is also the frequency for "error" beeps (for example, if you press an illegal key). To disable all error beeps set this or BeepLength 107 to 0. If you do, the BEEP 208 command will still be operable, but will not produce sound unless you explicitly specify the frequency and duration.

See also: BeepLength 107 and BEEP 208.

See other Configuration Directives 99.

5.2.14 BeepLength

BeepLength = *nnn*

Default: 2

Specifies the default BEEP 208 length in system clock ticks (approximately 1/18 of a second per tick). BeepLength is also the default length for "error" beeps (for example, if you press an illegal key)..

See also: BeepFreq 107 and BEEP 208.

See other Configuration Directives 99.

5.2.15 BeginLine

BeginLine = *Key*

Default: Home

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

See other General Input Keys 103.

5.2.16 BGColorRGB

BGColorRGB = *red,green,blue*

(TC) Set the default background color as an RGB value. The acceptable values for red, green, and blue are 0 to 255.

See other Configuration Directives 99.

5.2.17 CDDWinColors

CDDWinColors = *Color*

Sets the default colors for the popup window used by extended directory searches 14. If this directive is not used, the colors will be reversed from the current colors on the screen.

See other Color Directives 98.

5.2.18 CDDWinHeight

CDDWinHeight = *nnn*

Default: 16

Range: 5 to window rows - 3

Specifies, in characters, the initial height of the popup window used by extended directory searches 14. The value you enter will be adjusted if necessary to keep a minimum-size window visible on the screen.

See also: CDDWinLeft 108, CDDWinTop 108, and CDDWinWidth 109.

5.2.19 CDDWinLeft

CDDWinLeft = *nnn*

Default: 3

Specifies, in characters, the initial position of the left border of the popup window used by extended directory searches 14.

See also: CDDWinHeight 108, CDDWinTop 108, and CDDWinWidth 109.

5.2.20 CDDWinTop

CDDWinTop = *nnn*

Default: 3

Specifies, in characters, the initial position of the top border of the popup window used by extended directory searches 14.

See also: CDDWinHeight 108, CDDWinLeft 108, and CDDWinWidth 109.

5.2.21 CDDWinWidth

CDDWinWidth = *nnn*

Default: 72

Range: 10 to the window character size - 2

Specifies, in characters, the initial width of the popup window used by extended directory searches 14. The value you enter will be adjusted if necessary to keep a minimum-size window visible on the screen.

See also: CDDWinHeight 108, CDDWinLeft 108, and CDDWinTop 108.

5.2.22 ClearKeyMap

ClearKeyMap

Clears all current key mappings 102. ClearKeyMap is a special directive which has no value or = after it. Use ClearKeyMap with caution - it deletes all of the default definitions, and also any definitions in your *.INI* file directives that are processed before ClearKeyMap is processed. It is useful only if you want to make available most of the keys which have default assignments for other purposes, e.g., for keystroke aliases. ClearKeyMap should appear before any other key mapping directives. You may restore default mappings to keys you want to retain using the appropriate key assignment directives, e.g., NextFile 133=***Tab***.

To clear the default mappings for just a few keys, use the NormalEditKey 134, NormalKey 134, NormalListKey 134, or NormalPopupKey 134 directives, as appropriate to the command processor's input mode when you want the alternate operation available.

See other Advanced Directives 98.

5.2.23 CMDExtensions

CMDExtensions = YES | no

Enables or disables CMD.EXE-style command extensions. This only affects the MD 305 command (defaulting to /S); ***4NT*** and ***TC*** always enable all of the other extensions.

See other Configuration Directives 99.

5.2.24 ColorDIR (directive)

ColorDir = *ext1 [ext2 [...]]:colora[;ext3 [ext4 [...]]:colorb[; ...]]*

Sets the directory colors used by DIR 233 and SELECT 345 ***(4NT)***. The format is the same as that used for the COLORDIR 398 environment variable. See Color-Coded Directories 238 for a detailed explanation. You can use **and** / **or** / **xor** tests for the attributes.

See other Color Directives 98.

5.2.25 CommandEscape

CommandEscape = *Key*

Default: Alt-255

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

See other Command Line Editing Keys 103.

5.2.26 CommandSep

CommandSep = *c*

Default: &

Invalid characters: You cannot use any of the redirection 36 characters (| > <) or any of the white space characters (space, tab, comma, or equal sign).

This is the character used to separate multiple commands on the same line. It can be dynamically modified by the /C option of the SETDOS 354 command.

See the %+ 410 internal variable and the section on Special Character Compatibility 72 for information on using compatible command separators for two or more products.

The command separator is saved by SETLOCAL 358 and restored by ENDLOCAL 256.

See other Configuration Directives 99.

5.2.27 CompleteHidden

CompleteHidden = yes | NO

Yes includes and **No** excludes hidden and system files and directories from the list of names that can be returned for filename completion 58.

See other Configuration Directives 99.

5.2.28 CompletePaths

CompletePaths = yes | NO

If set to **YES**, filename completion will search for all matching files in the directories in the PATH 399 environment variable.

See other Configuration Directives 99.

5.2.29 ConsoleColumns

ConsoleColumns = *nnn*

Default: 80

Range: 80 ... 256

(*TC*) Sets the width of the ***TC*** console 86 screen buffer used for console applications. This directive controls the text buffer used for the console window, not the actual width of the window on your screen. The window width is determined by the operating system, not ***TC***, and may be smaller than the text buffer width. If it is, a horizontal scroll bar is provided to allow viewing of all text columns.

See other Initialization Directives 101.

5.2.30 ConsoleRows

ConsoleRows = *nnnn*

Default: 100

Range: 25 ... 8192

(*TC*) Sets the height of the ***TC*** console 86 screen buffer used for console applications. This directive controls the text buffer used for the console window, not the actual height of the window on your screen. The window height is determined by the operating system, not ***TC***, and may be smaller than the text buffer height (if it is, a vertical scroll bar is provided to allow viewing of all text rows).

See other Initialization Directives 101.

5.2.31 CopyPrompt

CopyPrompt = yes | NO

If set to **Yes**, the command processor will prompt in COPY 216 and MOVE 308 before overwriting an existing file if the command is being performed at the command prompt. This duplicates the behavior of newer versions of CMD.EXE.

See other Configuration Directives 99.

5.2.32 CUA

CUA = yes | NO

(TC) With the default setting of **No**, ***TC*** will use the command Windows (non-CUA) editing keys: **Ctrl-X** for cut, **Ctrl-C** for copy, and **Ctrl-V** for paste. If set to **Yes**, ***TC*** will use the **Common User Access** (CUA) standard keys for cut and paste: **Ctrl-Del** for cut, **Ctrl-Ins** for copy, and **Shift-Ins** for paste.

See other Configuration Directives 99.

5.2.33 Copy (directive)

Copy = *Key*

Default: Ctrl-Y

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

See other General Input Keys 103.

5.2.34 CursorIns

CursorIns = *nnnn*

Default: 15 ***(TC)*** 100 ***(4NT)***

This is the shape of the cursor for insert mode during command line editing and all commands which accept line input (DESCRIBE, ESET, etc.). The size is a percentage of the total character cell size, between 0% and 100%. Because of the way video drivers map the cursor shape, you may not get a smooth progression in cursor shapes as **CursorIns** and **CursorOver** change. If you set CursorIns and CursorOver to -1, the cursor shape won't be modified at all. If you set them to 0, the cursor will be invisible. See also: CursorOver 112, SETDOS /S 354.

See other Configuration Directives 99.

5.2.35 CursorOver

CursorOver = *nnn*

Default: 100 ***(TC)*** 15 ***(4NT)***

This is the shape of the cursor for overstrike mode during command line editing and all commands which accept line input. The size is a percentage of the total character cell size, between 0% and 100%. For more details see the CursorIns 112 directive; also see SETDOS 354 /S.

See other Configuration Directives 99.

5.2.36 Debug

Debug = *n*

Default: 2

Controls certain debugging options which can assist you in tracking down unusual problems. Use the following values for **Debug** (to enable more than one option, add the desired values together):

0 Disabled.
1 During the startup process, display the complete command tail passed to the command processor, then wait for a keystroke.
2 (default) Include the product name with every error message displayed by the command processor. This may be useful if you are unsure of the origin of a particular error message.

See also: the batch file debugger 201, a separate and unrelated facility for stepping through batch files.

See other Advanced Directives 98.

5.2.37 DebuggerTransparency

DebuggerTransparency = *nnn*

Default: 255

nnn must be in the range **40..255** and specifies the default transparency of the debugger windows. You can also set it interactively with the slider control on the bottom left of the debugger window.

5.2.38 DecimalChar

DecimalChar = . | , | AUTO

Sets the character used as the decimal separator for @EVAL 451, numeric IF and IFF tests, version numbers, and other similar uses. The only valid settings are period [**.**], comma [**,**], and **Auto** (the default). A setting of **Auto** tells the command processor to use the decimal separator associated with your current country code. If you change the decimal character you must also adjust the thousands character (with ThousandsChar 147) so that the two characters are different. See also: SETDOS /G 354 .

See other Configuration Directives 99.

5.2.39 Del (directive)

Del = *Key*

Default: Del

Deletes the character at the cursor.

See other General Input Keys 103.

5.2.40 DelayedExpansion

DelayedExpansion = yes | NO

Enables or disables CMD.EXE-style delayed expansion. Setting DelayedExpansion=YES is the same as specifying /V on the command processor startup line.

See other Configuration Directives 99.

5.2.41 DelGlobalQuery

DelGlobalQuery = YES | no

Enable or disable the confirmation prompt from DEL /Q when doing a wildcard-only or directory deletion. Use caution if you set DelGlobalQuery to **No**, as this will allow DEL /Q to delete an entire directory without prompting for confirmation. See DEL 225 for additional details.

See other Configuration Directives 99.

5.2.42 DelHistory

DelHistory = *Key*

Default: Ctrl-D

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

See other Command Line Editing Keys 103.

5.2.43 DelToBeginning

DelToBeginning = *Key*

Default: Ctrl-Home

Deletes from the cursor to the start of the line.

See other General Input Keys 103.

5.2.44 DelToEnd

DelToEnd = *Key*

Default: Ctrl-End

Deletes from the cursor to the end of the line.

See other General Input Keys 103.

5.2.45 DelWordLeft

DelWordLeft = *Key*

Default: Ctrl-L

Deletes the word to the left of the cursor.

See other General Input Keys 103.

5.2.46 DelWordRight

DelWordRight = *Key*

Default: Ctrl-R, Ctrl-Bksp

Deletes the word to the right of the cursor. See ClearKeyMap 109 if you need to remove the default mapping of Ctrl-Bksp to this function.

See other General Input Keys 103.

5.2.47 DescriptionMax

DescriptionMax = *nnn*

Default: 511

Controls the description length limit for DESCRIBE 230. The allowable range is 20 to 511 characters.

See other Configuration Directives 99.

5.2.48 DescriptionName

DescriptionName = *File*

Sets the file name in which to store file descriptions. The default file name is DESCRIPT.ION. See also: SETDOS 354 /D.

See other Configuration Directives 99.

5.2.49 Descriptions

Descriptions = YES | no

Turns description handling on or off during the file processing commands. If set to No, the command processor will not update the description file when files are moved, copied, deleted or renamed. See also: SETDOS 354 /D.

See other Configuration Directives 99.

5.2.50 DirHistory

DirHistory = *nnnn*

Default: 1024

Sets the amount of memory allocated to the directory history list (in characters). The allowable range is 1,024 to 32767. If you use a global directory history list (see Local and Global History Lists 52), the DirHistory value is ignored in all sessions except that which first establishes the global list. (To change the size, you will need to close all of the ***4NT*** and ***TC*** sessions, including SHRALIAS 361.)

See other Initialization Directives 101.

5.2.51 DirWinOpen

DirWinOpen = *Key*

Default: Ctrl-PgUp, F6

Opens the directory history window while at the command line. **(*TC*)** See also: Scrolling and History Keystrokes 51.

See other Popup Window Keys 104.

5.2.52 Down

Down = *Key*

Default: Down

Scrolls the display down one line in LIST 298; moves the cursor down one line in SELECT 345 and in the command line history, directory history, or @SELECT 489 window.

See other General Input Keys 103.

5.2.53 DuplicateBugs

DuplicateBugs = yes | NO

Tells the parser to duplicate certain well-known bugs in CMD.EXE. The only bug currently replicated is in the IF 286 command.

See other Initialization Directives 101.

5.2.54 EditMode

EditMode = Insert | Overstrike| InitOverstrike | InitInsert

This directive lets you start the command line editor in either **Insert** (***TC*** default) or **Overstrike** (***4NT*** default) mode. If you specify InitOverstrike or InitInsert, the command line editor will start in the specified state, but if you toggle insert mode while editing a line, the editor will continue to use the new mode on subsequent lines. See also: SETDOS 354 /M.

EditMode defaults to Insert in ***TC***, and to Overstrike in ***4NT***.

See other Configuration Directives 99.

5.2.55 Editor

Editor = *File*

(***TC***) Specifies the path and filename of the program that ***TC*** will execute when you select "Editor" from the Utilities menu 77, from LIST 298, or in the FIND 262 dialog. The default is *NOTEPAD.EXE*. (***4NT*** supports **Editor** in LIST 298.)

See other Configuration Directives 99.

5.2.56 EndHistory

EndHistory = *Key*

Default: Ctrl-E

Displays the last entry in the history list.

See other Command Line Editing Keys 103.

5.2.57 EndLine

EndLine = *Key*

Default: End

Moves the cursor to the end of the line.

See other General Input Keys 103.

5.2.58 EraseLine

EraseLine = *Key*

Default: Esc

Deletes the entire line.

See other General Input Keys 103.

5.2.59 EscapeChar

EscapeChar = *c*

Sets the character used to suppress the normal meaning of the following character. The default is a caret (**^**). See Escape Character 69 for a description of the special escape sequences. You cannot use any of the redirection 36 characters (|, >, or <) or the white space characters (space, tab, comma, or equal sign) as the escape character. The escape character is saved by SETLOCAL 358 and restored by ENDLOCAL 256. See also: SETDOS 354 /E. See the %= 410 internal variable and the section on Special Character Compatibility 72 for information on using compatible escape characters for two or more products.

See other Configuration Directives 99.

5.2.60 EvalMax

EvalMax = *nnnn*

Default: 10

Controls the maximum number of digits after the decimal point in values displayed by @EVAL 451. You can override this setting with the construct @EVAL[expression=n,n]. The allowable range is 0 to 10. See also: SETDOS 354 /F.

See other Configuration Directives 99.

5.2.61 EvalMin

EvalMin = *nnnn*

Default: 0

Controls the minimum number of digits after the decimal point in values displayed by @EVAL 451. The allowable range is 0 to 10. This directive will be ignored if EvalMin is larger than EvalMax. You

can override this setting with the construct @EVAL[expression=n,n]. See also: SETDOS 354 /F.

See other Configuration Directives 99.

5.2.62 ExecLine

ExecLine = *Key*

Default: Enter

Executes or accepts a line.

See other General Input Keys 103.

5.2.63 ExecWait

ExecWait = yes | NO

Controls whether the command processor waits for an external program started from the command line to complete before redisplaying the prompt. See Waiting for Applications to Finish 68 for details on the effects of this directive.

See other Configuration Directives 99.

5.2.64 ExitFile

ExitFile = YES | no

If set to **NO**, don't attempt to execute *4EXIT* ***(4NT)*** or *TCEXIT* ***(TC)*** when exiting the command processor.

See other Configuration Directives 99.

5.2.65 FGColorRGB

FGColorRGB = *red,green,blue*

(TC) Set the default background color as an RGB value. The acceptable values for red, green, and blue are 0 to 255.

See other Configuration Directives 99.

5.2.66 FileCompletion (directive)

FileCompletion = *cmd1: ext1 ext2 ...; cmd2: ext3 ext4 ...*

Sets the files made available during filename completion for selected commands. The format is the same as that used for the FILECOMPLETION 399 environment variable. See Customizing Filename Completion 60 for a detailed explanation of selective filename completion.

See other Configuration Directives 99.

5.2.67 FirewallHost

FirewallHost = *name*

The server name of the firewall, if any, to use for FTP and HTTP access.

See FirewallPassword 119, FirewallType 119, FirewallUser 119 and other Configuration Directives 99.

5.2.68 FirewallPassword

FirewallPassword = *password*

The password for FirewallUser 119 if required for authentication by the firewall.

Note: This directive stores the password in plain text, making it accessible to anyone who can read your *.INI* file.

See FirewallHost 119, FirewallType 119 and other Configuration Directives 99.

5.2.69 FirewallType

FirewallType = None | Tunnel | SOCKS4 | SOCKS5

See FirewallPassword 119, FirewallHost 119, FirewallUser 119 and other Configuration Directives 99.

5.2.70 FirewallUser

FirewallUser = *name*

The user name if the firewall requires authentication. A password can be defined via the FireWallPassword 119 directive

See FirewallHost 119, FirewallType 119 and other Configuration Directives 99.
.

5.2.71 FTPCFG

FTPCFG = *filename*

Specify the location and name of the file containing the FTP user names and passwords, and optionally the directory format for non-standard FTP servers. The default is **FTP.CFG** in the ***4NT*** or ***TC*** directory.

We recommend you encrypt this file if you're using NTFS. If FTP.CFG doesn't exist the first time ***4NT*** or ***TC*** looks for it, it will be created as an encrypted file (NTFS only). If you are using FAT/VFAT, the file will be in plain unencrypted text.

See Using FTP/HTTP Servers 42 for details.

5.2.72 FTPTimeout

FTPTimeout = *nnn*

Default: 120

Set the timeout (inactivity) period in seconds for FTP / FTPS operations.

See other Configuration Directives 99.

5.2.73 FuzzyCD

FuzzyCD = 0 | 1 | 2 | 3

Enables or disables extended directory searches, and controls their behavior. A setting of **0** (the default) disables extended searches. For complete details on the meaning of the other settings see Extended Directory Searches 14.

See other Configuration Directives 99.

5.2.74 Help (directive)

Help = *Key*

Default: F1

Displays the Help File 506 topic for the current command. See also: the HELP 284 command and the HelpWord 120 directive.

See other Command Line Editing Keys 103.

5.2.75 HelpWord

HelpWord = *Key*

Default: Ctrl-F1

Invokes the HELP 506 facility for the word at the cursor.

See other Command Line Editing Keys 103.

5.2.76 HideConsole

HideConsole = YES | no

(TC) Normally ***TC*** will hide the console window 86 after running a console mode program. If HideConsole is set to No, the console window will only be hidden until the first character-mode application you run. After that it remains on-screen. (The console window will be visible but will not remain "on top". You can make it the top window, or return the ***TC*** window to the top, by pressing **Alt-V**). This option works on all screens, but is primarily intended for high-resolution screens where you can resize the console and ***TC*** windows to run side by side.

See other Initialization Directives 101.

5.2.77 HistCopy

HistCopy = yes | NO

Controls what happens when you re-execute a line from the command history. If this option is set to **Yes**, the line is appended to the end of the history list. By default, or if this option is set to **No**, the command is not copied. The original copy of the command is always retained at its original position in the list, regardless of the setting of HistCopy. Set this option to **No** if you want to use HistMove 122 = **Yes**; otherwise, the HistCopy setting will override HistMove.

See other Configuration Directives 99.

5.2.78 HistDups

HistDups = OFF | first | last

Controls duplicate entry placement in the history list 49.

Off	Add new entry unconditionally.
First	Add new entry only if it does not match any old entries
Last	Add new entry unconditionally. Delete matching older entries.

See other Configuration Directives 99.

5.2.79 HistFile

HistFile = *File*

If HistFile is set, the command processor will load the specified history list file (via history /r filename) before running 4START, and save the command history to it (history > filename) after running 4EXIT.

See other Initialization Directives 101.

5.2.80 HistLogName

HistLogName = *File*

Sets the history log file name and path. If this directive is not used, the default name of ***TCHLOG*** **(*TC*** **)** or ***4NTHLOG*** **(*4NT*)** in the root directory of the boot drive will be used. Using HistLogName does not turn history logging on. To do so, you must use a LOG 303 /H ON command or the HistLogOn 121 directive. See also: LogName 131.

See other Configuration Directives 99.

5.2.81 HistLogOn

HistLogOn = yes | NO

If set to **Yes**, history logging 303 is turned on when the command processor starts.

See other Configuration Directives 99.

5.2.82 HistMin

HistMin = *nnnn*

Default: 0

Sets the minimum command line size to save in the command history list. Any command line whose length is less than this value will not be saved. Legal values range from **0**, which saves everything, to **1024**. You can prevent any command line from being saved in the history by beginning it with an "at" sign ("@"). See also:: HISTORY 49 command.

See other Configuration Directives 99.

5.2.83 HistMove

HistMove = yes | NO

If set to **Yes**, a recalled line is moved to the end of the command history. The difference between this directive and HistCopy, 121 is that HistCopy = Yes copies each recalled line to the end of the history but leaves the original in place. **HistMove = Yes** places the line at the end of history and removes the original line. This directive has no effect if **HistCopy = Yes**.

See other Configuration Directives 99.

5.2.84 History

History = *nnnn*

Default: 8192

Sets the amount of memory allocated to the command history list (in characters). The allowable range of values is from 4000 to 131071. If you use a global history list 52, the History value is ignored in all shells except the shell which first establishes the global list. (To change the size, you will need to close all of the 4NT and TC sessions, including SHRALIAS 361.)

See other Initialization Directives 101.

5.2.85 HistWinOpen

HistWinOpen = *Key*

Default: PgUp for ***4NT***, Ctrl-PgUp for ***TC***

Brings up the history window while at the command line. **(*TC*)** See also: Scrolling and History Keystrokes 51 and the SwapScrollKeys 145 directive.

See other Popup Window Keys 104.

5.2.86 HistWrap

HistWrap = YES | no

Controls whether the command history "wraps" when you reach the top or bottom of the list. The default setting enables wrapping, so the list appears "circular". If HistWrap is set to **No**, history recall will stop at the beginning and end of the list rather than wrapping.

See other Configuration Directives 99.

5.2.87 HTTPTimeout

HTTPTimeout = *nnn*

Default: 120

Set the timeout (inactivity) period in seconds for HTTP / HTTPS operations.

See other Configuration Directives 99.

5.2.88 IBeamCursor

IBeamCursor = YES | no

(TC) If set to **Yes**, ***TC*** will display the standard "I-Beam" cursor in text areas of its window. If IBeamCursor is set to **No**, the "Arrow" cursor is used in all areas of the window. This can be helpful on laptop systems where the I-Beam cursor is hard to see. You may need to restart ***TC*** to see the changes.

The cursors are those defined in your Windows configuration (generally under the "Pointers" tab of the "Mouse Properties" dialog).
IBeamCursor=YES uses the cursor referred to as "I-Beam" or "Text Select", and **IBeamCursor=No** uses the cursor referred to as "Arrow" or "Normal Select".

See other Initialization Directives 101.

5.2.89 InactiveTransparency

InactiveTransparency = *nnn*

(TC) Set the transparency level of the Take Command window when it is out of focus. The range is 40-255.

See other Configuration Directives 99.

5.2.90 Include

Include = *File*

Include the text from the named file at this point in the processing of the current *.INI* file. Use this option to share a file of directives between JP Software products. The text in the named file is processed just as if it were part of the original *.INI* file. When the include file is finished, processing resumes at the point where it left off in the original file. The included file may contain any valid directive for the current section, but may not contain a section name. Includes may be nested up to three levels deep (counting the original file as level 1). You must maintain include files manually —

the configuration dialogs modify the original *.INI* file only, and do not update included files.

See other Advanced Directives 98.

5.2.91 INIQuery

INIQuery = yes | NO

If set to **Yes**, a prompt will be displayed before execution of each subsequent line in the current *.INI* file. This allows you to modify certain directives when you start the command processor in order to test different configurations. INIQuery can be reset to **No** at any point in the file. Normally INIQuery = Yes is only used during testing of other *.INI* file directives.

The dialog displayed by ***TC*** when INIQuery = Yes gives you three options:

Yes Executes the directive
No Skips the directive
Cancel Executes the directive and all remaining directives in the [TakeCommand] section of the *.INI* file (*i.e.*, cancels the INIQuery = Yes setting)

The ***4NT*** prompt generated by INIQuery = Yes is:

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

At this prompt, you may enter:

Y Process this line and go on to the next.
N Skip this line and go on to the next.
Q Skip this line and all subsequent lines.
R Execute this and all subsequent lines.
E Prompt for a new value for this entry.

If you choose **E**, you can enter a new value for the directive, but not a new directive name.

See other Initialization Directives 101.

5.2.92 InputColors

InputColors = *Color*

Sets the colors used for command line input. This setting is useful for making your input stand out from the normal output.

See other Color Directives 98.

5.2.93 Ins

Ins = *Key*

Default: Ins

Toggles insert / overstrike mode during line editing.

See other General Input Keys 103.

5.2.94 JabberPassword

JabberPassword = *password*

Set the default password to use when logging onto a Jabber server and sending IM's via the JABBER 294 command.

See other Initialization Directives 101.

5.2.95 JabberServer

JabberServer = *servername*

Set the default server to use when sending IM's via the JABBER 294 command.

See other Initialization Directives 101.

5.2.96 JabberUser

JabberUser = *username*

Set the default user name to use when logging onto a Jabber server and sending IM's via the JABBER 294 command.

See other Initialization Directives 101.

5.2.97 LastHistory

LastHistory = *Key*

Default: F3

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

See other Command Line Editing Keys 103.

5.2.98 Left

Left = *Key*

Default: left arrow, ←

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

See other General Input Keys 103.

5.2.99 LFNToggle

LFNToggle = *Key*

Default: Ctrl-A

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

See other Command Line Editing Keys 103.

5.2.100 LineToEnd

LineToEnd = *Key*

Default: Ctrl-Enter

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

See other Command Line Editing Keys 103.

5.2.101 ListBack

ListBack = *Key*

Default: B

Returns to the previous file.

See other LIST Keys 104.

5.2.102 ListboxBarColors

ListboxBarColors = *Color*

(4NT) Sets the color for the highlight bar in the popup list boxes (i.e., command history window 51, filename completion window 61, @SELECT window 489, etc.).

See other Color Directives 98.

5.2.103 ListClipboard

ListClipboard = *Key*

Default: Ctrl-B

Copy the current LIST 298 filename to the clipboard

See other LIST Keys 104.

5.2.104 ListColors

ListColors = *Color*

Sets the colors used by the LIST 298 command. If this directive is not used, LIST will use the current default colors set by the CLS 215 or COLOR 216 command or by the StdColors 145 directive.

See other Color Directives 98.

5.2.105 ListContinue

ListContinue = *Key*

Default: C

Go to the next file.

See other LIST Keys 104.

5.2.106 ListExit

ListExit = *Key*

Default: Esc

Exits from the LIST 298 command.

See other LIST Keys 104.

5.2.107 ListFind

ListFind = *Key*

Default: F

Prompts and searches for a string.

See other LIST Keys 104.

5.2.108 ListFindRegex

ListFindRegex = *Key*

Default: R

Perform a regular expression search in LIST 298.

See other LIST Keys 104.

5.2.109 ListFindRegexReverse

ListBack = *Key*

Default: Ctrl-R

Perform a backwards regular expression search in LIST 298.

See other LIST Keys 104.

5.2.110 ListFindReverse

ListFindReverse = *Key*

Default: Ctrl-F

Prompts and searches backward for a string.

See other LIST Keys 104.

5.2.111 ListHex

ListHex = *Key*

Default: X

Toggles between hexadecimal and character display modes.

See other LIST Keys 104.

5.2.112 ListHighBit

ListHighBit = *Key*

Default: H

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

See other LIST Keys 104.

5.2.113 ListInfo

ListInfo = *Key*

Default: I

Displays information about the current file.

See other LIST Keys 104.

5.2.114 ListInverseColors

ListInverseColors = *fg* on *bg*

(4NT) Set the color to use in LIST when showing search matches.

See other Configuration Directives 99.

5.2.115 ListNext

ListNext = *Key*

Default: N

Finds the next matching string.

See other LIST Keys 104.

5.2.116 ListOpen

ListOpen = *Key*

Default: O

Opens the common Windows "open file" dialog to select a new file to LIST 298.

See other LIST Keys 104.

5.2.117 ListPrevious

ListPrevious = *Key*

Default: Ctrl-B

Finds the previous matching string.

See other LIST Keys 104.

5.2.118 ListPrint

ListPrint = *Key*

Default: P

Prints the file on the default printer.

See other LIST Keys 104.

5.2.119 ListRefresh

ListRefresh = *Key*

Default: F5

Refresh the LIST 298 display. (Useful when viewing a growing log file.)

See other LIST Keys 104.

5.2.120 ListRowStart

ListRowStart = 1 | 0

Specifies whether LIST 298 and FFIND 262 consider the first line in a file to be line **1** or line **0**. The default is **1**.

See other Configuration Directives 99.

5.2.121 ListStatBarColors

ListStatBarColors = *Color*

(4NT) Sets the colors used on the LIST 298 status bar. If this directive is not used, LIST 298 will set the status bar to the reverse of the screen color (the screen color is controlled by ListColors 126).

See other Color Directives 98.

5.2.122 ListUnicode

ListUnicode = *Key*

Default: U

Toggles the LIST 298 display mode between Unicode and ASCII.

See other LIST Keys 104.

5.2.123 ListWrap

ListWrap = *Key*

Default: W

Toggles LIST 298's wrap option on and off. The wrap option wraps text at the right margin.

See other LIST Keys 104.

5.2.124 LocalAliases

LocalAliases = yes | NO

The default value is **No,** which forces all copies of the command processor to share the same alias list. **Yes** keeps the lists for each shell separate. See ALIAS 187 for more details on local and global alias lists.

See other Initialization Directives 101.

5.2.125 LocalDirHistory

LocalDirHistory = yes | NO

The default value is **No**, which forces all copies of the command processor to share the same directory history list. **Yes** keeps the list for each session separate. See Directory History Window 62 for an explanation of local and global directory histories.

See other Initialization Directives 101.

5.2.126 LocalFunctions

LocalFunctions = yes | NO

The default value is **No**, which forces all copies of the command processor to share the same function list. **Yes** keeps the lists for each shell separate. See FUNCTION 275 for more details on local and global function lists.

See other Initialization Directives 101.

5.2.127 LocalHistory

LocalHistory = yes | NO

The default value is **No**, which forces all copies of the command processor to share the same history list. **Yes** keeps the lists for each shell separate. See Local and Global History Lists 52 for more details on local and global history lists.

See other Initialization Directives 101.

5.2.128 LogAll

LogAll = yes | NO

(TC) If set to YES, Take Command will save everything that is written to the Take Command window to the log file.

See other Configuration Directives 99.

5.2.129 LogErrors

LogErrors = yes | NO

If set to **Yes**, error messages will be written to the log file. See the LOG 303 command for additional information.

See other Configuration Directives 99.

5.2.130 LogName

LogName = *File*

Sets the log file name and path. If this directive is not used, the default name of ***TCLOG* (*TC*)** or ***4NTLOG* (*4NT*)** in the root directory of the boot drive will be used. Using LogName does not turn logging on; you must use a LOG 303 ON command or the LogOn 131 directive to do so. See also: HistLogName 121.

See other Configuration Directives 99.

5.2.131 LogOn

LogOn = yes | NO

If set to **Yes,** command logging 303 is turned on when the command processor starts.

See other Configuration Directives 99.

5.2.132 MailAddress

MailAddress = *name*

The email address of the current user, used in SENDMAIL 349 for outgoing mail . If not set, SENDMAIL will attempt to get the address from the registry. This value is what will be transmitted in the "*From:*" header of messages you send.

See MailPassword 132, MailPort 132, MailServer 132, MailUser 132 and other Configuration Directives 99.

5.2.133 MailPassword

MailPassword = *string*

The email password of the current user (MailUser 132), used in SENDMAIL 349 for outgoing mail if the mail server requires it. If not set, SENDMAIL will attempt to get the password from the registry.

Note: This directive stores the password in plain text, making it accessible to anyone who can read your *.INI* file.

See MailAddress 132, MailPort 132, MailServer 132, MailUser 132 and other Configuration Directives 99.

5.2.134 MailPort

MailPort = *n*

The SMTP port number (the default is **25**) for use by SENDMAIL 349.

See MailAddress 132, MailPassword 132, MailServer 132, MailUser 132 and other Configuration Directives 99.
.

5.2.135 MailServer

MailServer = *name*

The local SMTP server name to use in SENDMAIL 349 for outgoing mail. If not set, SENDMAIL will attempt to get the address from the registry.

See MailAddress 132, MailPassword 132, MailPort 132, MailUser 132 and other Configuration Directives 99.

5.2.136 MailUser

MailUser = *name*

The email username of the current SENDMAIL 349 user if your SMTP server 132 requires it for authentication. If a password is also required, it can be defined by the MailPassword 132 directive.

See MailAddress 132, MailPassword 132, MailPort 132, MailServer 132 and other Configuration Directives 99.

5.2.137 MouseWheel (4NT)

MouseWheel = YES | no

NO disables mouse wheel support in LIST 298 in ***4NT*** (to reduce interference with other applications).

5.2.138 NextFile

NextFile = *Key*

Default: F9, Tab

Gets the next matching filename during filename completion. See ClearKeyMap 109 if you need to remove the default mapping of Tab to this function.

See other Command Line Editing Keys 103.

5.2.139 NextHistory

NextHistory = *Key*

Default: Down for ***4NT***, Ctrl-Down for ***TC***

Recalls the next command from the command history. **(*TC*)** See also: Scrolling and History Keystrokes 51 and the SwapScrollKeys 145 directive.

See other Command Line Editing Keys 103.

5.2.140 NextINIFile

NextINIFile = *File*

The full path and name of the file must be specified. All subsequent shells will read the specified *.INI* file, and ignore any [**Secondary**] section in the original *.INI* file.

See other Advanced Directives 98

5.2.141 NoClobber

NoClobber = yes | NO

If set to **Yes**, will prevent standard output redirection 36 from overwriting an existing file, and will require that the output file already exist for append redirection. See also: SETDOS 354 /N.

You can override NoClobber by adding the exclamation point to the output redirection symbol. For example:

```
dir >! dir.out
```

See other Configuration Directives 99.

5.2.142 NormalEditKey

NormalEditKey = *Key*

Deassigns a command line editing key in order to disable the usual meaning of the key while editing a command line, and/or make it available for keystroke aliases. This will make the keystroke operate as a "normal" key with no special function. See NormalKey 134 for an example.

5.2.143 NormalKey

NormalKey = *Key*

Deassigns a general input key in order to disable the usual meaning of the key within the command processor and/or make it available for keystroke aliases. This will make the keystroke operate as a "normal" key with no special function. For example:

```
NormalKey = Ctrl-End
```

will disable Ctrl-End, which is the standard "delete to end of line" key. Ctrl-End could then be assigned to a keystroke alias. Another key could be assigned the "delete to end of line" function with the DelToEnd 114 directive.

See other General Input Keys 103.

5.2.144 NormalListKey

NormalListKey = *Key*

Deassigns a LIST 298 key in order to disable the usual meaning of the key within LIST. This will make the keystroke operate as a "normal" key with no special function. See NormalKey 134 for an example.

See other LIST Keys 104.

5.2.145 NormalPopupKey

NormalPopupKey = *Key*

Deassigns a popup window key in order to disable the usual meaning of the key within the popup window. This will make the keystroke operate as a "normal" key with no special function. See NormalKey 134 for an example.

See other Popup Window Keys 104.

5.2.146 NTFSDescriptions

NTFSDescriptions = yes | NO

If **YES**, use the Comments field in the NTFS SummaryInformation stream for each file to hold its description, instead of the *DESCRIPT.ION* file. The advantages are that the description will always remain with the file regardless of what program copies, moves, or renames it. The disadvantages are

that accessing stream objects in Windows is slow, and you cannot attach a description to directories.

See other Initialization Directives 101.

5.2.147 ParameterChar

ParameterChar = *c*

Sets the character used after a percent sign to specify all or all remaining command line parameters in a batch file or alias (e.g., **%$** or **%n$**; see Batch File Parameters 165 and ALIAS 187). The default is the dollar sign [**$**]. The parameter character is saved by SETLOCAL and restored by ENDLOCAL. See also: SETDOS 354 /P. See Special Character Compatibility 72 for information on using compatible parameter characters for two or more products.

See other Configuration Directives 99.

5.2.148 PassiveFTP

PassiveFTP = YES | no

Passive FTP mode is usually required if you have a firewall. You should only set PassiveFTP to **NO** if you are having FTP connection problems, or if you must use the PORT command instead of the PASV command.

This setting applies to both the IFTP 288 command and standalone ftp references 42. It can be temporarily changed when desired with the OPTION 317 command:

```
OPTION //passiveftp=no
IFTP ...
...
OPTION //passiveftp=yes
```

See other Configuration Directives 99.

5.2.149 Paste

Paste = *Key*

Default: Ctrl-V

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

See other General Input Keys 103.

5.2.150 PathExt (directive)

PathExt = yes | NO

Determines whether the command processor will use the PATHEXT environment variable. If set to **No** (the default), the PATHEXT variable is ignored. If set to **Yes**, the PATHEXT variable will be used to determine extensions to look for when searching the PATH for an executable file. For details, see the PATHEXT 399 variable and the PATH 319 command.

Caution: If you set PathExt = Yes in the *.INI* file and then fail to set the PATHEXT variable, path

searches will fail as there will be no extensions for which to search!

PATHEXT is supported for compatibility reasons but should not generally be used as a substitute for the more flexible executable extensions 33 feature.

See other Configuration Directives 99.

5.2.151 PauseOnError

PauseOnError = YES | no

(4NT) **Yes** forces a pause with the message "Error in *filename*, press any key to continue processing" after displaying any error message related to a specific line in the *.INI* file. **No** continues processing with no pause after an error message is displayed.

See other Initialization Directives 101.

5.2.152 Perl

Perl = yes | NO

If **YES**, enable the internal Perl 175 support. (This can only be set at startup, not for the current session with the OPTION command.)

See other Initialization Directives 101.

5.2.153 PopFile

PopFile = *Key*

Default: F7, Ctrl-Tab

Opens the filename completion window. You may not be able to use Ctrl-Tab, because not all systems recognize it as a keystroke. See ClearKeyMap 109 if you need to remove the default mapping of Ctrl-Tab to this function.

See other Command Line Editing Keys 103.

5.2.154 PopupWinBegin

PopupWinBegin = *Key*

Default: Ctrl-PgUp

Moves to the first item in the list when in the popup window.

See other Popup Window Keys 104.

5.2.155 PopupWinColors

PopupWinColors = *Color*

(4NT) Sets the default colors for the command line, directory history, and filename completion

windows, and most other popup windows (see CDDWinColors 108 for the extended directory search window). If this directive is not used, the colors will be reversed from the current colors on the screen.

See other Color Directives 98.

5.2.156 PopupWinDel

PopupWinDel = *Key*

Default: Ctrl-D

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

See other Popup Window Keys 104.

5.2.157 PopupWinEdit

PopupWinEdit = *Key*

Default: Ctrl-Enter

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

See other Popup Window Keys 104.

5.2.158 PopupWinEnd

PopupWinEnd = *Key*

Default: Ctrl-PgDn

Moves to the last item in the list when in the popup window.

See other Popup Window Keys 104.

5.2.159 PopupWinExec

PopupWinExec = *Key*

Default: Enter

Selects the current item and closes the window.

See other Popup Window Keys 104.

5.2.160 PopupWinHeight

PopupWinHeight = *nnnn*

Specifies, in characters, the initial height of the Command History Window 51 and of the Directory History Window 62. The value you enter will be adjusted if necessary to keep a minimum-size window

visible on the screen.

Default: 12
***Minimum*:** 5

See also: PopupwinLeft 138, PopupWinTop 138, PopupWinWidth 138..

5.2.161 PopupWinLeft

PopupWinLeft = *nnn*

Specifies, in characters, the initial position of the left border of the Command History Window 51 and of the Directory History Window 62.

Default: 40

See also: PopupWinHeight 137, PopupWinTop 138, PopupWinWidth 138..

5.2.162 PopupWinTop

PopupWinTop = *nnn*

Specifies, in characters, the initial position of the top border of the Command History Window 51 and of the Directory History Window 62.

Default: 1

See also: PopupWinHeight 137, PopupwinLeft 138, PopupWinWidth 138..

5.2.163 PopupWinWidth

PopupWinWidth = *nnn*

Specifies, in characters, the initial width of the Command History Window 51 and of the Directory History Window 62. The value you enter will be adjusted if necessary to keep a minimum-size window visible on the screen.

Default: 36
Minimum: 10

See also: PopupWinHeight 137, PopupwinLeft 138, PopupWinTop 138.

5.2.164 PrevFile

PrevFile = *Key*

Default: F8, Shift-Tab

Gets the previous matching filename. See ClearKeyMap 109 if you need to remove the default mapping of Shift-Tab to this function.

See other Command Line Editing Keys 103.

5.2.165 PrevHistory

PrevHistory = *Key*

Default: Up for ***4NT***, Ctrl-Up for ***TC***

Recalls the previous command from the command history. **(*TC*)** See also: Scrolling and History Keystrokes 51 and the SwapScrollKeys 145 directive.

See other Command Line Editing Keys 103.

5.2.166 Proxy

Proxy = *name*

Specifies the proxy server address to use for HTTP calls.

See ProxyPassword 139, ProxyPort 139, ProxyUser 139 and other Configuration Directives 99.

5.2.167 ProxyPassword

ProxyPassword = *password*

Password for HTTP proxy if Basic authentication is to be used.

See Proxy 139, ProxyPort 139, ProxyUser 139 and other Configuration Directives 99.

5.2.168 ProxyPort

ProxyPort = *port*

Specifies the port number to use for HTTP calls.

See Proxy 139, ProxyPassword 139, ProxyUser 139 and other Configuration Directives 99.

5.2.169 ProxyUser

ProxyUser = *name*

A user name if Basic authentication is to be used for the HTTP proxy.

See Proxy 139, ProxyPassword 139, ProxyPort 139 Configuration Directives 99.

5.2.170 RecycleBin

RecycleBin = yes | NO

If set to **Yes**, files deleted by the DEL/ERASE 225 commands and by RD /S 333 are placed in the Windows Recycle Bin by default. If set to **No**, the files are deleted without being placed in the Recycle Bin. DEL 225's and RD 333's /K and /R switches allow you to override this setting for individual commands. The RecycleExclude 400 environment variable can be used to exclude specific files.

See other Configuration Directives 99.

5.2.171 RegularExpressions

RegularExpressions = *type*

Sets the type of regular expression syntax to use. The options are:

PERL
Ruby
Java
grep
POSIX
GNU

The default value is **PERL**.

See other Initialization Directives 101.

5.2.172 RepeatFile

RepeatFile = *Key*

Default: F12

Repeats the previous matching filename during filename completion.

See other Command Line Editing Keys 103.

5.2.173 REXX

REXX = YES | no

If **YES**, enable the internal REXX 175 support. (This can only be set at startup, not for the current session with the OPTION command.)

See other Initialization Directives 101.

5.2.174 Right

Right = *Key*

Default: Right

Moves the cursor right one character on the input line; scrolls the display right 8 columns in LIST 298; scrolls the display right 4 columns in the command line history, directory history, or @SELECT 489 window.

See other General Input Keys 103.

5.2.175 RLocalHost

RLocalHost = *name*

Set the name of the local host or user-assigned IP interface through which connections are initiated or accepted (for REXEC and RSHELL).

See other Initialization Directives 101.

5.2.176 RLocalPort

RLocalPort = *n*

Set the port number (or communication endpoint) in the local machine to bind to for REXEC 340 and RSHELL 341.

See other Initialization Directives 101.

5.2.177 RLocalUser

RLocalUser = *name*

Set the name of the user on the local machine (for RSHELL 341).

See other Initialization Directives 101.

5.2.178 Ruby

Ruby = yes | NO

If **YES**, enable the internal Ruby 175 support. (This can only be set at startup, not for the current session with the OPTION command.)

See other Initialization Directives 101.

5.2.179 SaveDirCase

SaveDirCase = YES | no

Enables or disables special processing to maintain the original case of each path element when changing directories. This processing is necessary for a few programs which are case-sensitive in their use of directory names. If you do not use such a program, disabling this case preservation will speed up directory changes slightly.

See other Advanced Directives 98.

5.2.180 SaveHistory

SaveHistory = *Key*

Default: Ctrl-K

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

See other Command Line Editing Keys 103.

5.2.181 ScreenBufSize

ScreenBufSize = *nnnn*

Default: 200,000

(TC) Sets the size of the screen scrollback buffer (in characters). The allowable range is from 50,000 to 1,000,000.

See other Initialization Directives 101.

5.2.182 ScreenColumns

ScreenColumns = *nnnn*

(TC) Sets the number of virtual screen columns used by the ***TC*** window. If the virtual screen width is greater than the physical window width, ***TC*** will display a horizontal scrollbar at the bottom of the window. See Resizing the Take Command Window 84 for more information on the virtual screen size.

See other Configuration Directives 99.

5.2.183 ScreenRows

ScreenRows = *nnn*

Default: 25

(TC) Sets the initial height of the ***TC*** window. See Resizing the Take Command Window 84 for more information on screen size.

See other Configuration Directives 99.

5.2.184 ScrollDown

ScrollDown = *Key*

(TC) Scrolls the command processor's scrollback buffer 83 down one line.

See other Scrollback Buffer Keys 104.

5.2.185 ScrollPgDn

ScrollPgDn = *Key*

(TC) Scrolls the command processor's scrollback buffer 83 down one page.

See other Scrollback Buffer Keys 104.

5.2.186 ScrollPgUp

ScrollPgUp = *Key*

(TC) Scrolls the command processor's scrollback buffer 83 up one page.

See other Scrollback Buffer Keys 104.

5.2.187 ScrollUp

ScrollUp = *Key*

(TC) Scrolls the command processor's scrollback buffer 83 up one line.

See other Scrollback Buffer Keys 104.

5.2.188 SelectColors

SelectColors = *Color*

Sets the colors used by the SELECT 345 command. If this directive is not used, SELECT will use the current default colors set by the CLS 215 or COLOR 216 command or by the StdColors 145 directive.

See other Color Directives 98.

5.2.189 SelectStatBarColors

SelectStatBarColors = *Color*

(4NT) Sets the color used on the SELECT 345 status bar. If this directive is not used, SELECT will set the status bar to the reverse of the screen color (the screen color is controlled by SelectColors 143).

See other Color Directives 98.

5.2.190 ServerCompletion

ServerCompletion = none | LOCAL | global

Specifies how server name completion should proceed (see Filename Completion 58 for information on how to use server name completion). The default of **Local** lists only local servers (i.e., those in your "network neighborhood"). **Global** will enumerate the entire network. **None** will disable server completion; this may be necessary to prevent "hanging" if you start typing a server name and accidentally press Tab, and your local domain is very large or slow to respond.

See other Configuration Directives 99.

5.2.191 SettingChange

SettingChange = yes | NO

If set to **YES**, the command processor will monitor the WM_SETTINGCHANGE message and if the environment is specified, update the environment from the User, Volatile, and System registry entries. The updates are done whenever the command processor displays a prompt (to prevent the environment from changing in the middle of a batch file).

See other Initialization Directives 101.

5.2.192 SHChangeNotify

SHChangeNotify = yes | NO

If set to **NO**, don't notify the system shell when 4NT / TC change files or directories. (This is faster but could introduce significant incompatibilities, particularly in future versions of Windows.)

The shell notification is done by the ASSOC 197, COPY 216, DEL 225, MD 305, MOVE 308, and RD 333 commands. Note that setting ShChangeNotify=YES could introduce a slight incompatibility with CMD.EXE, which doesn't notify the system shell about anything.

5.2.193 SSLPort

SSLPort = *port*

The port number for the FTP **SSL** service. The default is **21**. For implicit SSL, use port **990**.

See SSLProvider 144, SSLStartMode 144 and other Configuration Directives 99

5.2.194 SSLProvider

SSLProvider = *string*

The name of the security provider to use for SSL authentication. The default value is "*****", which selects the default SSL provider defined in the system. In current Windows implementations, this is " Microsoft Unified Security Protocol Provider".

See SSLPort 144, SSLStartMode 144 and other Configuration Directives 99

5.2.195 SSLStartMode

SSLStartMode = *n*

Specify how to start SSL negotiation:

0 (default) If the remote port is set to the standard plaintext port of the protocol, SSL will behave the same as if SSLStartMode is set to 2 (sslExplicit). In all other cases, SSL negotiation will be implicit (sslImplicit).

1 (sslImplicit) The SSL negotiation will start immediately after the connection is established.

2 (sslExplicit) Connection will first be in plaintext, and then SSL negotiation will be explicitly started through a protocol command such as STARTTLS.

3 (sslNone) No SSL negotiation, no SSL security. All communication will be in plaintext mode.

See SSLPort 144, SSLProvider 144 and other Configuration Directives 99

5.2.196 StartupFile

StartupFile = YES | no

If set to **NO**, don't attempt to execute 4START 8 ***(4NT)*** or TCSTART 8 ***(TC)*** at startup.

See other Configuration Directives 99.

5.2.197 StatusBarOn

StatusBarOn = YES | no

(TC) **Yes** enables the status bar when ***TC*** starts; **No** disables it. The status bar can be enabled or disabled while ***TC*** is running by using the Options menu 77. The StatusBarOn setting is automatically updated to reflect the current state of the status bar each time ***TC*** exits; this preserves the status bar state between sessions.

See other Configuration Directives 99.

5.2.198 StatusBarText

StatusBarText = *nnnn*

Default: 8

(TC) Sets the size of the text on the status bar, in points. The allowable range is 4 to 16. Correct operation is not guaranteed for other values.

See other Configuration Directives 99.

5.2.199 StdColors

StdColors = *Color*

(4NT) Sets the standard colors to be used when CLS 215 is used without a color specification. Using this directive is similar to placing a COLOR 216 command in the 4START 8 file.

(TC) Sets the standard colors to be used when CLS 215 is used without a color specification. Using this directive is similar to placing a COLOR 216 command in the TCSTART 8 file.

See other Color Directives 98.

5.2.200 SwapScrollKeys

SwapScrollKeys = yes | NO

(TC) **YES** switches to ***4NT***-style keystrokes for manipulating the scrollback buffer 83.

If SwapScrollKeys is set to **Yes**, the Up and Down arrow keys will scroll through the command history 49 list and the PgUp key will pop up the history window 51. The Ctrl-Up, Ctrl-Down, Ctrl-PgUp, and Ctrl-PgDn keys will scroll the text in the screen buffer.

If SwapScrollKeys is set to **No**, these keys will assume their default meanings. The Up and Down arrow keys and the PgUp and PgDn keys will scroll the text in the screen buffer. The Ctrl-Up and Ctrl-Down keys will scroll through the command history list and the Ctrl-PgUp key will pop up the history window.

For additional details see Scrolling and History Keystrokes 51.

Note: Do not set SwapScrollKeys to Yes if you use key mapping directives 102 to reassign the scrolling or history keys individually. SwapScrollKeys takes effect before other key mappings, and

using both methods at the same time will be confusing at best. Setting SwapScrollKeys to Yes has essentially the same effect as including the following key mapping directives in the .INI file 91 individually:

```
PrevHistory = Up
NextHistory = Down
HistWinOpen = PgUp
HistWinOpen = PgDn
ScrollUp = Ctrl-Up
ScrollDown = Ctrl-Down
ScrollPgUp = Ctrl-PgUp
ScrollPgDn = Ctrl-PgDn
```

See other Configuration Directives 99.

5.2.201 SwitchChar

SwitchChar = *c*

Default: /

Set the switch character. Note that this will not affect any external applications, and will probably break all of your existing batch files and aliases!

See other Configuration Directives 99.

5.2.202 TabStops

TabStops = *nn*

Default: 8

(TC) Sets the tab stops for the command processor's output (including the output from the LIST 298 and TYPE 384 commands). ***(4NT)*** Sets the tab stops for LIST 298 output. The allowable range is 1 to 32.

See other Configuration Directives 99.

5.2.203 TCStartPath

TCStartPath = *Path*

(TC) Sets the drive and directory where ***TC*** looks for the TCSTART and TCEXIT 8 batch files (if any).

See other Initialization Directives 101.

5.2.204 TFTPTimeout

TFTPTimeout = *nn*

Default: 120

Set the timeout (inactivity) period in seconds for TFTP operations.

See other Configuration Directives 99.

5.2.205 ThousandsChar

ThousandsChar = . | , | AUTO

Sets the character used as the thousands separator for numeric output. The only valid settings are period [**.**], comma [**,**], and **Auto** (the default). **Auto** tells the command processor to use the thousands separator associated with your current country code. If you change the thousands character you must also adjust the decimal character (with DecimalChar 113) so that the two characters are different. See also: SETDOS /G 354.

See other Configuration Directives 99.

5.2.206 TimeServer

TimeServer = *name*

Specifies the URL of the internet time server to use for TIME /S. If no server is specified, the TIME 378 command uses **clock.psu.edu**.

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

See other Initialization Directives 101.

5.2.207 ToolBarOn

ToolBarOn = YES | no

(TC) **YES** enables the tool bar when ***TC*** starts; **NO** disables it. The tool bar can be enabled or disabled while ***TC*** is running by using the Options menu 77. The ToolBarOn setting is automatically updated to reflect the current state of the tool bar each time ***TC*** exits; this preserves the tool bar state between sessions.

See other Configuration Directives 99.

5.2.208 ToolBarText

ToolBarText = *nn*

Default: 8

(TC) Sets the point size of text on the tool bar. The allowable range is 4 to 16.

See other Configuration Directives 99.

5.2.209 Transparency

Transparency = *nn*

Default: 255

(TC) Set the default transparency level of the Take Command window. The range is 0 (invisible) to 255 (opaque).

See other Configuration Directives 99.

5.2.210 TreePath

TreePath = *Path*

Sets the location of *JPSTREE.IDX*, the file used for the extended directory search 14 database. By default, the file is placed in the root directory of drive **C:**.

TreePath does not change the NAME of the index file, only its location.

See other Initialization Directives 101.

5.2.211 UnicodeOutput

UnicodeOutput = NO | yes

Disable / enable Unicode output for redirection. This option will be overridden by a **/U** or **/A** startup command line option 4. It can also be modified dynamically with the OPTION 317 command.

.

5.2.212 UnixPaths

UnixPaths = yes | NO

Enables the forward slash as a path separator in the command name (the first item on the command line). This allows you to enter a command such as:

```
c:\> /bin/programs/xyz.exe
```

without having the forward slashes interpreted as switch characters. Note that setting UnixPaths to Yes does not change the command processor or Windows switch character, it's still **/**. It simply allows you to put forward slashes in the command name without problems. See SwitchChar 146 if you want to change the switch character.

When UnixPaths is set to Yes command switches beginning with a forward slash must be preceded by a space to avoid confusion (this is a good general practice regardless of the setting of UnixPaths). For example:

```
c:\> \bin\xyz.exe /c        OK
c:\> /bin/xyz.exe /c        OK
c:\> \bin\xyz.exe/c         Error
c:\> /bin/xyz.exe/c         Error
```

See other Configuration Directives 99.

5.2.213 Up

Up = Key

Default: Up

Scrolls the display up one line in LIST 298; moves the cursor up one line in SELECT 345 and in the command line history, directory history, or @SELECT 489 window.

See other General Input Keys 103.

5.2.214 UpdateTitle

UpdateTitle = YES | no

The command processor normally changes the title in its title bar to include the command or batch file name each time a new command is executed. If you prefer a static title bar which does not change with each command, set UpdateTitle to **NO** to prevent these updates.

See other Configuration Directives 99.

5.2.215 VariableExpand

VariableExpand = *Key*

Default: Ctrl-X

Expands variables at the command prompt.

See other Command Line Editing Keys 103.

5.2.216 Win32SFNSearch

Win32SFNSearch = yes | NO

Set to **YES** to if you want to search for both short filenames and long filenames. See LFN File Searches 31 for details.

See other Configuration Directives 99.

5.2.217 WindowHeight

WindowHeight = *nnn*

If WindowState 149 is set to **CUSTOM**, it specifies - in pixels 569 - the initial height of the command processor window.

See also: WindowWidth 150, WindowX 150, WindowY 150.

5.2.218 WindowState

WindowState = STANDARD | maximize | minimize | hidden | custom

Sets the initial state of the command processor window. **Standard** puts the window in the default position on the Windows desktop, and is the default setting. **Maximize** maximizes the window; **Minimize** minimizes it, **Hidden** hides it (from the desktop and the taskbar), and **Custom** sets it to the position specified by the WindowX, WindowY, WindowWidth, WindowHeight 150 directives.

See other Initialization Directives 101.

5.2.219 WindowWidth

WindowWidth = *nnnn*

If WindowState 149 is set to **CUSTOM**, it specifies - in pixels 569 - the initial width of the command processor window.

See also: WindowHeight 149, WindowX 150, WindowY 150.

5.2.220 WindowX

WindowX = *nnnn*

If WindowState 149 is set to **CUSTOM**, it specifies - in pixels 569 - the initial position of the left border of the command processor window.

See also: WindowHeight 149, WindowWidth 150,, WindowY 150.

5.2.221 WindowY

WindowY = *nnnn*

If WindowState 149 is set to **CUSTOM**, it specifies - in pixels 569 - the initial position of the top border of the command processor window.

See also: WindowHeight 149, WindowWidth 150, WindowX 150.

5.2.222 WordLeft

WordLeft = *Key*

Default: Ctrl-Left

Moves the cursor left one word; scrolls the display left 40 columns in LIST 298.

See other General Input Keys 103.

5.2.223 WordRight

WordRight = *Key*

Default: Ctrl-Right

Moves the cursor right one word; scrolls the display right 40 columns in LIST 298.

See other General Input Keys 103.

5.2.224 Wow64FsRedirection

Wow64FSRedirection = YES | no

If set to NO, will override the default Win64 behavior of remapping windows\system32 calls to windows\SysWOW64.

See other Configuration Directives 99.

5.2.225 ZoneID

ZoneID = *n*

Default: 0

If set to a non-zero value, it enables the XP / NTFS ZoneId security when running executables downloaded from the Internet. The default value is **0**; when enabling it you should normally set it to **3**. (Note that ***CMD.EXE*** never checks for the ZoneID, so setting it may introduce a minor incompatibility.)

See other Configuration Directives 99.

5.3 Configuration Dialog

This dialog, available via the OPTION 317 command, and also from the Options menu 77 in ***TC***, contains several "pages" or "tabs" of options that let you change the way the command processor looks and works. Each option sets a corresponding directive in the .INI file 91 which was used to start the command processor.

Unless you select the **Cancel** button, any changes you make will take effect immediately. If you select **Apply**, the settings will only apply for the duration of that session. If you select **Save**, the settings will be recorded in the appropriate main section (**[4NT]** or **[TC]**) of the .INI file 91 and will be in effect each time you start that command processor. If you want to set configuration directives in the **[Primary]** or **[Secondary]** sections of the *.INI* file, you must edit the file directly instead of using the dialog. Similarly, if you modified directives that originally resided in a separate included 123 INI file, new directives will be saved to the main *.INI* file but the included file itself will not be altered. It would be wise to verify that no "old" directive in included files override the changes you made into the main file.

For details about the *.INI* file and *.INI* file directives, the allowable ranges for each, and the effect of each, see Initialization (.INI) Files 88 and Directives 91.

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

Startup 152
Windows 153
Editing 154
Colors 154
History 155
Syntax 156
Miscellaneous 157
Internet 156
Email 156
Batch Debugger 157
Caveman 158 ***(TC)***
Registration 158

5.3.1 Startup Options Tab

If you are not familiar with the purpose or use of the configuration dialogs, review the main configuration dialogs 151 topic before continuing. If you need more details about a particular option, see the Initialization directive related to that option.

Using the top entry, you can set the path to your 4START / 4EXIT 8 (***4NT***) or TCSTART / TCEXIT 8 (***TC***) files if they aren't in the same directory as the command processor. This field sets the 4StartPath 105 or TCStartPath 146 directive as appropriate.

Local History instructs the command processor to use a local or (if unchecked) global command history list (LocalHistory 131 directive).

Local Aliases instructs the command processor to use a local or (if unchecked) global alias list (LocalAliases 130 directive).

Local Directory History instructs the command processor to use a local or (if unchecked) global directory history (LocalDirHistory 130 directive).

Local Functions instructs the command processor to use a local or (if unchecked) global function list (LocalFunctions 131 directive).

SFN Search enables or disables a search of short file names after long filenames on LFN drives (Win32SFNSearch 149 directive).

PathExt enables or disables the use of the PATHEXT environment variable (PathExt 135 directive). Note that setting this variable but failing to create the PATHEXT environment variable will cause PATH searches to fail.

Delete to Recycle Bin determines whether deleted files are placed in the Recycle Bin (RecycleBin 139 directive).

Default batch echo sets the default batch echo state (BatchEcho 107 directive).

Copy prompt on overwrite determines whether the command processor will prompt in COPY or MOVE before overwriting an existing file if the command is being performed at the command prompt (CopyPrompt 111 directive).

Protect redirected output files determines whether files are protected from being overwritten during output redirection 36 (NoClobber 133 directive).

Wait for external apps determines whether or not the command processor waits for external applications to end (ExecWait 118 directive).

Update Titles prevents or allows updating of the command processor window title (UpdateTitle 149 directive).

Unix-style Paths enables or disables forward slashes in paths (UnixPaths 148 directive).

Zone ID enables or disables the XP / NTFS ZoneId security (ZoneID 151 directive).

In the Logging section:

Errors determines whether error messages are written to the log file (LogErrors 131 directive).

Command turns command logging on by default (LogOn 131 directive). If you enter a file name in the File field, that file will be used for logging and the LogName 131 directive will be set.

History turns history logging on by default (HistLogOn 121 directive). If you enter a file name in the File field, that file will be used for history logging that the HistLogName 121 directive will be set.

In the Scripting section:

REXX enables or disables the internal REXX 175 support.

Perl enables or disables the internal Perl 175 (ActiveState 5.8) support.

Ruby enables or disables the internal Ruby 175 (1.8) support.

Note : you must restart the command processor for the **REXX**, **Perl**, and **Ruby** options to take effect.

5.3.2 Window Options Tab

If you are not familiar with the purpose or use of the configuration dialogs, review the main configuration dialogs 151 topic before continuing. If you need more details about a particular option, see the Initialization directive related to that option.

In the Display section:

The **Standard**, **Max**, **Min**, and **Custom** radio buttons select the initial state for the command processor windows (WindowState 149 directive).

The **X**, **Y**, **Width**, and **Height** fields set the initial size and position of the command processor window (WindowX, WindowY, WindowWidth, and WindowHeight 150 directives). They are ignored unless the **Custom** radio button is also selected.

(TC) The **Transparency** field sets the transparency level for the Take Command window (Transparency 148 directive).

In the Text section:

The **Tabs** field sets the tabs stops for the LIST 298 command's output (TabStops 146 directive).

(TC) The **Width** option sets the virtual screen width (ScreenColumns 142 directive).

(TC) The **Height** option sets the virtual screen height (ScreenRows 142 directive).

In the Window Configuration section:

(TC) **Enable Toolbar** sets the tool bar mode at startup (ToolBarOn 147 directive). The **Font Size** field sets the point size of tool bar text (ToolBarText 147 directive). The tool bar will only be visible if you have defined one or more tool bar buttons 79.

(TC) **Enable Statusbar** sets the status bar mode at startup (StatusBarOn 145 directive). The **Font Size** field sets the point size of status bar text (StatusBarText 145 directive).

In the Cursor section:

(TC) The **Arrow** and **I-Beam** radio buttons let you select the type of cursor which ***TC*** will use (

IBeamCursor 123 directive).

***(TC)* In the Screen Buffer section:**

(TC) The **Buffer Size** field sets the number of bytes of memory allocated to the ***TC*** window buffer (ScreenBufSize 141 directive).

In the Console section:

(TC) The **Rows** and **Columns** options set the height and width of the console-mode screen buffer (ConsoleRows 111 and ConsoleColumns 110 directives).

(TC) The **Hide** option determines whether the console window is hidden after a character-mode application ends (HideConsole 120 directive).

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

5.3.3 Editing Options Tab

If you are not familiar with the purpose or use of the configuration dialogs, review the main configuration dialogs 151 topic before continuing. If you need more details about a particular option, see the Initialization directive related to that option.

In the Editing section:

The **Default** radio buttons set the start-up editing mode (EditMode 116 directive). The "Insert" and "Overstrike" modes are returned to their original settings whenever you begin to edit a new line; the "InitInsert" and "InitOverstrike" modes are not.

(TC) The **Edit keys** select the keystrokes used for Cut, Copy and Paste (CUA 111 directive).

The **Cursor** fields set the cursor size for both overstrike and insert editing modes (CursorOver 112 and CursorIns 112 directives). ***(TC)*** These options have no effect if you have selected an Arrow cursor on the Windows Options tab. 153

In the Filename Completion section:

The **Complete hidden files** option selects whether hidden and system files and directories can be returned by filename completion (CompleteHidden 110 directive).

The **Add '\' to Directories** option selects whether '\' is automatically appended to directory names (or '/' to FTP URLs) in filename completion (AppendToDir 106 directive).

The **Search PATH** option selects whether the directories in the PATH variable are searched if a match isn't found in the current directory (CompletePaths 110 directive).

The **Options** field lets you select file types available for filename completion (FileCompletion 118 directive).

5.3.4 Colors Options Tab

If you are not familiar with the purpose or use of the configuration dialogs, review the main configuration dialogs 151 topic before continuing. If you need more details about a particular option, see the Initialization directive related to that option.

The **Enable ANSI** option enables ***4NT***'s and ***TC***'s ANSI X3.64 support (ANSI 105 directive).

The options in the **Colors** section let you select foreground and background colors for input and normal output in the command processor window. These options set the InputColors 124 and StdColors 145 directives.

The options in the **List Colors** section select the foreground and background colors for the LIST window (ListColors 126 directive).

The options in the **SELECT Colors** section set colors used by the SELECT 345 command (SelectColors 143 directive).

The **DIR Colors** field lets you set the directory colors used by DIR and (in ***4NT)***, SELECT (ColorDir 109 directive).

5.3.5 History Options Tab

If you are not familiar with the purpose or use of the configuration dialogs, review the main configuration dialogs 151 topic before continuing. If you need more details about a particular option, see the Initialization Directives 101 related to that option.

1. The **Buffer Sizes** fields set the size of the **Command History** and **Directory History** buffers (History 122 and DirHistory 115 directives).

2. The **Pop-Up Windows** fields set the size and position of :

   - the command and directory **History** search windows (PopupWinLeft, PopupWinTop, PopupWinWidth, PopupWinHeight 137 directives), and

   - the special **Directory** popup window used by extended directory searches 14 (CDDWinLeft, CDDWinTop, CDDWinWidth, CDDWinHeight 108 directives).

3. In **Command History** section:

   - **(*TC*)** The **Scroll / History Keys** radio buttons select the set of keys used to manipulate the scrollback buffer and the history list. Before changing these settings, be sure to read about the SwapScrollKeys 145 directive, which is set by this option.

   - The **Minimum Saved** field sets the minimum command length to save in the command history (HistMin 122 directive).

   - The **Copy to end**, **Move to end**, and **Wrap** options select how the command history operates. Before changing these settings, be sure to read about the HistCopy 121, HistMove 122 and HistWrap 123 directives.

   - The **Duplicates** option selects what the command history does with duplicate entries (HistDups 121 directive).
   - The **HistoryFile** option loads the specified history list file at startup (HistFile 121 directive).

5.3.6 Syntax Options Tab

If you are not familiar with the purpose or use of the configuration dialogs, review the main configuration dialogs 151 topic before continuing. If you need more details about a particular option, see the Initialization directive related to that option.

In the **Special Characters** section:

The **Separator**, **Escape**, and **Parameter** options let you select the command separator character, escape character, and parameter character. Please see the CommandSep 110, EscapeChar 117, and ParameterChar 135 directives, which are set by these options, for details about these three characters, especially if you need to share batch files and aliases between 2 or more of our command processors.

The **Decimal** radio buttons select the character used as the decimal separator (Auto selects the separator designated by your country code) (DecimalChar 113 directive).

The **Thousands** radio buttons select the character used as the thousands separator (Auto selects the separator designated by your country code) (ThousandsChar 147 directive).

The radio buttons in the **Country** section select how time values are displayed by the command processor (AmPm 105 directive).

The two **Default Beep** options set the length (in clock ticks) and frequency for the command processor's "error" beeps (BeepLength 107 and BeepFreq 107 directives).

5.3.7 Internet Options Tab

If you are not familiar with the purpose or use of the configuration dialogs, review the main configuration dialogs 151 topic before continuing. If you need more details about a particular option, see the Initialization directive related to that option.

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

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

Time Server is the internet time server to use to set the time (TimeServer 147 directive). If no server is specified, TIME uses clock.psu.edu.

Passive FTP sets passive mode for FTP calls (sometimes required by a firewall) (PassiveFTP 135 directive).

The **Timeouts** section set the timeout (inactivity) period in seconds for FTP (FTPTimeout 119 directive), TFTP (TFTPTimeout 146 directive), and HTTP (HTTPTimeout 123 directive).

5.3.8 Email Options Tab

If you are not familiar with the purpose or use of the configuration dialogs, review the main configuration dialogs 151 topic before continuing. If you need more details about a particular option, see the Initialization directive related to that option.

In the **SMTP** section, **Server** is the local SMTP server name to use in SENDMAIL for outgoing mail. (If not set, SENDMAIL will attempt to get the address from the registry.) **Address** is the email

address of the current user, used in SENDMAIL for outgoing mail. (If not set, SENDMAIL will attempt to get the address from the registry.) **Port** is the port number to use for SMTP (the default is 25). **User** is the user name and **Password** the password is the loon password.

In the **JABBER** section, **Server** is the JABBER server to log into, **User** is the user name, and **Password** is the logon password.

5.3.9 Miscellaneous Options Tab

If you are not familiar with the purpose or use of the configuration dialogs, review the main configuration dialogs 151 topic before continuing. If you need more details about a particular option, see the Initialization directive related to that option.

In the **EVAL** section, the **Min** and **Max** fields set the minimum and maximum number of digits after the decimal point in values returned by @EVAL 451 and set the EvalMin 117 and EvalMax 117 directives.

(TC) The **Editor** field selects the program to run for the "Editor" menu choice (Editor 116 directive).

In the **Descriptions** section, the **Enable** option enables or disables file descriptions (Descriptions 115 directive). The **Maximum Length** field sets the maximum text length of file descriptions and the DescriptionMax 115 directive. The **NTFS Descriptions** stores file descriptions in the NTFS SummaryInformation stream (NTFSDescriptions 134 directive).

In the **Extended Directory Search** section, the **Search Level** radio buttons select the Extended Directory Search 14 mode and set the FuzzyCD 120 directive. The **TreePath** field sets the path to the *JPSTREE.IDX* file which contains the extended directory search database (TreePath 148 directive).

5.3.10 Debugger Tab

If you are not familiar with the purpose or use of the configuration dialogs, review the main configuration dialogs 151 topic before continuing. If you need more details about a particular option, see the Initialization directive related to that option.

The **Transparency** option sets the transparency level of the debugger windows (40 to 255). 40 is almost invisible; 255 is opaque (no transparency).

The **Extensions** option sets the batch file extensions supported by the debugger. If you don't specify an extension when creating a new file, the debugger will use the first extension in the list. Extensions must be separated by a semicolon. The option will take effect the next time a debugger window is opened.

The **Auto Bracket Match** option turns on bracket matching - **( )**, **[ ]**, and **{ }**. The option will take effect the next time a debugger window is opened.

If the **Syntax Coloring option** is enabled, key words (as defined in *BATCH.BCP)* will be displayed in a different color.

If the **Auto Indent** option is enabled, a new line will be automatically indented to match the previous line. The option will take effect the next time a debugger window is opened.

The **Smart Indent** option will increase the indent for new lines if the command on the previous line matches one in the smart indent list in *BATCH.BCP*. The option will take effect the next time a debugger window is opened.

The **Case Fixing** option will automatically change the case of key words to match that in

BATCH.BCP. (Syntax Coloring must also be enabled so that Case Fixing can recognize the keywords.) The option will take effect the next time a debugger window is opened.

The **Show Line Numbers** option will display line numbers before each line. The option will take effect the next time a debugger window is opened.

The **Backup Files** option saves a backup copy of the file when you do a Save. The file will be saved with a *.BAK* extension. The option will take effect the next time a debugger window is opened.

The **Font** button invokes a font dialog to select the font style and size to use in the debugger 201 windows.

The **Foreground RGB** button invokes a color picker dialog to select the foreground text color for debugger 201 windows.

The **Background RGB** button invokes a color picker dialog to select the background text color for debugger 201 windows.

5.3.11 Registration Tab

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

The Register tab allows you to register ***4NT*** or ***Take Command***. When you purchase a new or upgrade copy of ***4NT*** or ***TC***, you will receive an email with your name and registration key. Enter the registration information exactly as you received it in the email (preferably by cutting & pasting). Remember to save your registration key in a safe place in case you need to reinstall. If you have lost your registration key, you can request a replacement by contacting JP Software at sales@jpsoft.com, or at one of the addresses listed at the start of this file.

The **Remove Registration** button removes your name and serial number from the computer.

5.3.12 Caveman Options Tab

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

If you are not familiar with the purpose or use of the configuration dialogs, review the main configuration dialogs 151 topic before continuing. If you need more details about a particular option, see the Initialization directive related to that option.

This tab enables or disables Caveman support, and selects which character-mode applications should be run automatically under Caveman. See Caveman 86 for complete information on ***TC***'s Caveman feature and the applications it supports.

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

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

To add an application to the list, click the **Add** button to open the Caveman Apps dialog. To change

an existing entry, select it and then click **Edit**, or double-click the entry, which will open the same dialog for that entry. **Delete** removes an entry from the list.

In the Caveman Apps dialog, enter either the name or the full path name of an executable file, e.g., `DISKCOPY.COM` or `C:\WINDOWS\COMMAND\DISKCOPY.COM`. If you use the full path name, the entry will apply only to that specific file. If you use just the file name, the entry will apply to any file of that name, regardless of its location. If you use both, the full path name entry will be used when you execute that specific file, and the file name entry will be used for other files of the same name. Long filenames should not be quoted when they are entered in the **Command** field. You can use the **Browse…** button to open a standard Windows file open dialog to find a specific file.

You can also enter a wildcard 19 filename into the dialog, e.g., *`*.COM`*. This will enable Caveman for all character-mode programs which match the wildcard. For example, `C:\WINDOWS\COMMAND\*.COM` would enable all *`.COM`* files in the named directory, and *`*.COM`* would enable all *`.COM`* files in all directories. Extended wildcards 19 may be used as well.

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

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

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

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

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

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

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

6 Aliases & Batch Files

Whenever you have a command (internal or external) that you need to execute often, one that's too complex to be dependably typed manually at the Command Line 46, 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 160
- Batch Files 162
- Special Character Compatibility 72

6.1 Aliases

Much of the power of ***4NT*** and ***TC*** 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. ***4NT*** and ***TC*** also support Directory Aliases 18, a shorthand way of specifying pathnames.

This section shows you some examples of the power of aliases. See the ALIAS 187 command for complete details about writing your own aliases. You can create aliases either from the command line, as described in this section, or ***(TC)*** with the Aliases dialog 81 which is available from the Utilities menu 77.

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 211 \.

Aliases can also create customized versions of commands. For example, the DIR 233 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 187 command for full details.

When an alias contains multiple commands, the commands are executed one after the other. However, if any of the commands runs an external Windows application (such as the fictitious

PLAYER.EXE shown above), you must be sure the alias will wait for the application to finish before continuing with the other commands. See Waiting for Applications to Finish 68 for additional details.

Aliases can be nested; that is, one alias can invoke another. For example, the alias above could also be written as:

```
alias play=e:\music\player.exe
alias music=`pushd c:\sounds & play & popd`
```

If you enter *MUSIC* as a command, the command processor executes the PUSHD 331 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 326 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 225 command to prompt before it erases a file:

```
alias del=*del /p
```

An asterisk ***** is used in front of the second DEL to tell the command processor to use the original internal command, not an alias. See Temporarily Disabling Aliases 190 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 230 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 230 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 233 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 166 for details. Alias examples in this section assume the ***4NT*** and ***TC*** default of ParameterChar=$. If you are using a different ParameterChar 135 you will need to edit the examples accordingly.

Parameters that are referred to in an alias, but which are missing on the command line, appear as empty strings inside the alias. For example, if you only put two parameters on the command line, any reference in the alias to **%3** or any higher-numbered parameter will be interpreted as an empty string.

The parameter **%n$** has a special meaning. The command processor interprets it to mean "the entire command line, from parameter ***n*** to the end." If ***n*** is not specified, it has a default value of **1**, so **%$** 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 187 and UNALIAS 386 commands for more information and examples.

6.2 Batch Files

A batch file is a file that contains a list of commands to execute. The command processor reads and interprets each line as if it had been typed at the keyboard. Like aliases 187, 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:

6.2.1 .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 303 command.

6.2.2 Using .BAT Files Under 4NT

In most cases under ***4NT*** your batch files will be stored as *.CMD* or *.BTM* files. However, you may also choose to use some *.BAT* files, especially if you are moving from Win98 to Win2000 / XP / 2003 / Vista. If you do, you need to be aware of the way ***4NT*** and ***TC*** execute *.BAT* files, which is slightly different from the method used by CMD.EXE.

CMD.EXE passes all *.BAT* files to Windows' DOS command processor, COMMAND.COM, for execution (yes, there is a skeletal DOS command processor in Windows). COMMAND.COM handles a few DOS-related commands, but passes most internal commands to a second copy of CMD.EXE so that they are executed in the Windows environment. This convoluted system allows you to load memory-resident DOS programs (TSRs), and run other programs which use them, all from the same *.BAT* file. However, it reduces performance for all *.BAT* files in order to support those rare files which load DOS TSRs under Windows.

4NT and ***TC*** do not use this system; they execute *.BAT* files directly, just like *.CMD* and *.BTM* files. This works better for most files, but may render DOS TSRs loaded from a *.BAT* file ineffective

because other commands in the file are not executed in a DOS-based environment.

In most cases this difference will not affect your *.BAT* files, because you will not be loading DOS TSRs in Windows. If you do need to load TSRs from *.BAT* files, we recommend that you obtain a copy of our DOS command processor, ***4DOS***, start it from your Windows desktop, and run the *.BAT* files from ***4DOS***. You could also use CMD.EXE, but of course the *.BAT* files then cannot use ***4DOS*** or ***4NT / TC*** features. While we do not generally recommend using ***4DOS*** under Windows 2000 / XP / 2003, it works well in this specific situation.

When invoking DOS programs from a ***4NT*** or ***TC*** batch file, we recommend that you enable the *CONFIG.NT* directive **NTCMDPROMPT** without which Windows tends to "forget" to return control to a calling 32-bit program (such as ***4NT*** and ***TC***) and may leave you at an unexpected COMMAND.COM prompt. *CONFIG.NT* typically resides in the Windows *SYSTEM32* directory. See your Windows documentation for additional information.

6.2.3 Echoing in Batch Files

By default, each line in a batch file is displayed or "echoed" as it is executed. You can change this behavior, if you want, in several different ways:

- Any batch file line that begins with an @ symbol will not be displayed.
- The display can be turned off and on within a batch file with the ECHO 253 OFF and ECHO 253 ON commands.
- The default setting can be changed with the SETDOS 354 /V command, the "Default Batch Echo" checkbox on the Startup tab 152 of the configuration dialogs, or the BatchEcho 107 directive.

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
```

The command processor also has a command line echo that is unrelated to the batch file echo setting. See ECHO 253 for details about both settings.

6.2.4 Special syntax for CMD.EXE compatibility

For compatibility with CMD.EXE, the command processor also supports additional syntax to qualify references to parameters of batch files and the control variable of the FOR 267 command when referenced by the ***command*** it executes. However, this syntax is usually best replaced by the more flexible Variable Functions 424.

CMD.EXE syntax	Expands to	Suggested replacement
%*	All parameters	**%$**
%~n	unquoted (")	**%@replace[%=",,%n]**
%~fn	Fully qualified name of ***%n***	**%@full[%n]**
%~dn	Drive letter portion of ***%n***	**%@left[2,%@full[%n]]**
%~pn	Full path (no drive letter) of ***%n***	**%@right[-2,%@path[%@full[%n]]]**
%~nn	Root name (no extension) of ***%n***	**%@name[%n]**

%~xn	File extension of ***%n***	**.%@ext[%n]**
%~sn	Fully qualified short name of ***%n***	**%@sfn[%n]**
%~an	File attributes of ***%n***	**%@attrib[%n]**
%~tn	File date and time of ***%n***	**%@filedate[%n] %@filetime[%n]**
%~zn	File size of ***%n***, bytes	**%@filesize[%n]**
%~$PATH:n	Full name of the first match for %***n*** in %PATH 399	**%@search[%n]**

Notes

In the special case where the parameter to a **%~** variable is **0**, e.g., **%~f0**, the returned file name will always include the extension, as it does under CMD.EXE.

%~$PATH:n returns an empty string if the file ***%n*** is not found in the path.

References qualified by the tilde **~** trigger an error message when used improperly, e.g. if attempting to display the size of a string parameter which is not the name of a file.

6.2.5 Batch File Parameters

Like aliases 187, user-defined functions 275 and application programs, batch files can examine the command line that is used to invoke them. The command tail (everything on the command line after the batch file or alias name) is separated into individual positional parameters (also called parameters or batch variables) by scanning for the spaces, tabs, and commas that separate them. For aliases and functions, a forward slash (**/**) triggers the beginning of a new parameter, e.g. the string **xyz/abc** is separated into parameters **foo** and **/abc**.

These parameters are numbered from **%1** to **%511**. **%1** refers to the first parameter on the command line, **%2** to the second, and so on. It is up to the batch file to determine the meaning of each parameter. You can use double quotes to pass spaces, tabs, commas, and other special characters in a batch file parameter; see Parameter Quoting 166 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 359
%n$	the command tail starting with parameter number ***n***, modified by SHIFT 359
%-n$	the command tail from parameter 1 to ***n*** - 1
%$	the complete command tail, modified by SHIFT 359
%*	the complete command tail, unmodified by SHIFT 359

For example, **%3$** means the third and all subsequent parameters. The values of **%#**, **%n$, %-n$**, and **%$** will change if you use the SHIFT 359 command. To emulate CMD.EXE, SHIFT 359 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 286 and IFF 287 commands.)

Batch files can also use environment variables 395, internal variables 402, and variable functions 424.

Batch file parameters may also use the special CMD.EXE compatibility syntax 164.

6.2.5.1 Parameter Quoting

As the command processor parses 70 the command line, it looks for the command separator 65, conditional commands 65 (**||** and **&&**), white space (spaces, tabs, and commas), percent signs **%** which indicate variables 395 or batch file 162 parameters to be expanded, and redirection and piping 35 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 the command processor 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 187 for details.

You can disable and reenable back quotes and double quotes with the SETDOS 354 /X command.

6.2.6 Using Environment Variables

Batch files can use environment variables 395, internal variables 402, variable functions 424, or user-defined functions 275. You can use these variables and functions to determine system status (*e.g.*, the CPU type), resource levels (*e.g.*, the amount of free disk space), file information (*e.g.*, the date and time a file was last modified), and other information (*e.g.*, the current date and time). You can also perform arithmetic operations (including date and time arithmetic), manipulate strings and substrings, extract parts of a filename, and read and write files.

To create temporary variables for use inside a batch file, just use the SET 351 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 388 command to remove these variables from the environment at the end of your batch file. You can use SETLOCAL 358 and ENDLOCAL 256 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 451 to add a number to a text string), you'll get an error message or a meaningless return value.

6.2.7 Batch File Commands

Some commands are particularly suited to batch file processing. Each command is explained in detail in the Command Reference 181. Here is a list of some of the commands you might find most useful:

ACTIVATE 186	activates another window
BEEP 208	produces a sound of any pitch and duration through the computer's speaker
BREAKPOINT 209	set a breakpoint in the batch debugger
CALL 209	executes one batch file from within another
CANCEL 211	terminates all batch file processing
CLS 215	clears the command processor screen
COLOR 216	sets the command processor display colors
DEBUGSTRING 225	send text to the debugger
DO 246	starts a loop. The loop can be based on a counter, or on a conditional expression, strings, or files. ENDDO terminates the loop
DRAWBOX 250	draws a box on the screen
DRAWHLINE 251	draws horizontal lines on the screen
DRAWVLINE 252	draws vertical lines on the screen
ECHO 253	sends text to the standard output device
ECHOS 255	sends text to the standard output device
ECHOERR 253	sends text to the standard error device
ECHOSERR 255	sends text to the standard error device
ENDLOCAL 256	restores the settings that were saved and allows specific variables to be exported (see SETLOCAL 358)
ENDTEXT 376	ends the block of text started with TEXT 376
EVENTLOG 259	writes a string to the Windows application event log
FOR 267	executes commands for each file that matches a set of wildcards, or each entry in a list
GOSUB 280	executes a subroutine inside a batch file (see RETURN 340).
GOTO 282	branches to a different location in the batch file
IF 286	execute commands based on a conditional expression
IFF 287	
INKEY 291	collects keyboard input and store it in environment variables
INPUT 293	collects keyboard input and store it in environment variables
JABBER 294	send an instant message (IM)
KEYSTACK 296	sends keystrokes to applications
LOADBTM 303	changes the batch file operating mode
MSGBOX 313	displays a dialog box with standard buttons like Yes, No, OK, and Cancel, and returns the user's selection
ON 315	initializes error handling for Ctrl-C / Ctrl-Break, or for program and command errors
OSD 318	Display floating text on the desktop
PAUSE 320	displays a message and waits for the user to press a key
PDIR 320	creates a customized DIR-like display of directory contents
PLAYAVI 324	plays Windows .AVI files

PLAYSOUND 325	plays Windows sound files
POSTMSG 327	send a message to a window
QUERYBOX 332	displays a dialog box for text input
QUIT 333	ends the current batch file and optionally returns an exit code
REM 336	places a remark in a batch file
RETURN 340	terminates a subroutine (see GOSUB 280)
SCREEN 342	positions the cursor on the screen and optionally prints a message at the new location
SCRPUT 344	displays a message in color
SENDMAIL 349	sends an email message
SETLOCAL 358	saves the current disk drive, default directory, environment, alias list, and special character settings (see ENDLOCAL 256).
SHIFT 359	changes the numbering of the batch file parameters
SMPP 362	sends messages using the SMPP protocol
SNPP 363	sends a message to an alphanumeric pager
START 364	starts another session or window
SWITCH 367	selects a group of statements to execute based on the value of a variable
TCTOOLBAR 374	changes the TC tool bar buttons
TEXT 376	displays a block of text (see ENDTEXT 376)
TIMER 379	starts or reads a stopwatch
TITLE 380	changes the window title
VSCRPUT 391	displays a vertical message in color
WMIQUERY 394	Query the Windows Management Instrumentation interface

These commands, along with the internal variables and variable functions, make the enhanced batch file language extremely powerful. Your copy of the command processor includes a sample batch file, in the file EXAMPLES.BTM, that demonstrates some of the things you can do with batch files.

6.2.8 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 the command processor or an application program is running, how the application, if any, was written, whether BREAK 209 is ON or OFF, and whether the ON BREAK 315 command is in use.

If the command processor 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 333 command; answering **A** is similar to the CANCEL 211 command.

Note: ***(TC)*** When the CUA 111 directive is set to **NO**, **Ctrl-C** is not available for interrupting commands. Use **Ctrl-Break** instead.

6.2.9 Detecting 4NT or Take Command

From a batch file, you can determine if ***4NT*** or ***TC*** is loaded by testing for the variable function @EVAL 451, with a test like this:

```
if "%@eval[2 + 2]%" == "4" echo %_cmdproc is loaded!
```

This test can never succeed in CMD.EXE. (Other variable functions could also be used for the same purpose.) The _CMDPROC 413 internal variable tells you which specific command processor is running.

6.2.10 Using Aliases in Batch Files

One way to simplify batch file programming is to use aliases to hide unnecessary detail inside a batch file. For example, suppose you want a batch file to check for certain errors, and display a message and exit if one is encountered. This example shows one way to do so:

```
setlocal
unalias *
setdos /e%=^ /c%=& /p%=$
alias error `echo. & echo ERROR: %$ & goto dispmenu`
alias fatalerror `echo. & echo FATAL ERROR: %$ & quit`
alias in `pushd %1 & %2$ & popd`
if not exist setup.btm fatalerror Missing setup file!
call setup.btm
cls
:dispmenu
text
          1. Word Processing
          2. Solitaire
          3. Internet
          4. Exit
endtext
echo.
inkey Enter your choice:  %%userchoice
switch %userchoice
case 1
   input Enter the file name:  %%fname
   if not exist fname error File does not exist
   in d:\letters c:\windows\wordpad.exe
case 2
   in d:\finance c:\windows\sol.exe
case 3
   in d:\comm c:\windows\iexplore.exe
case 4
   goto done
default
  error Invalid choice, try again
endswitch
goto dispmenu
:done
endlocal
```

The first alias, ERROR, simply displays an error message and jumps to the label DISPMENU to redisplay the menu. The **%$** in the second ECHO 253 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 331 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 367 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 211 to CDD 213 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 358, ENDLOCAL 256, and UNALIAS 386 to clear out aliases before your batch file starts, and SETDOS 354 to select the special characters you depend on, and restore them at the end, as we did in the previous example. Remember that SETLOCAL 358 and ENDLOCAL 256 will save and restore not only the aliases but also the environment, the current drive and directory, and various special characters.

If this method isn't appropriate or necessary for the batch file you're working on, you can also use an asterisk ***** before the name of any command. The asterisk means the command that follows it should not be interpreted as an alias. For example the following command redirects a list of file names to the file *FILELIST*:

```
dir /b > filelist
```

However, if the user has redefined DIR with an alias this command may not do what you want. To get around this just use:

```
*dir /b > filelist
```

The same can be done for any command in your batch file. If you use the asterisk, it will disable alias processing, and the rest of the command will be processed normally as an internal command, external command, or batch file. Using an asterisk before a command will work whether or not there is actually an alias defined with the same name as the command. If there is no alias with that name, the asterisk will be ignored and the command will be processed as if the asterisk wasn't there.

You can use the pseudovariables %= 410 and %+ 410 to represent the command escape 69 and command separator 110 characters, respectively, There is no pseudovariable for the parameter character 135.

6.2.11 Debugging Batch Files

4NT and ***TC*** include a built-in full-featured batch file debugger invoked with the BDEBUGGER 201 command. The debugger gives you a detailed, step-by-step view of batch file execution, and will help solve particularly difficult batch file problems.

6.2.12 String Processing

As you gain experience with batch files, you're likely to find that you need to manipulate text strings. You may need to prompt a user for a name or password, process a list of files, or find a name in a phone list. All of these are examples of string processing – the manipulation of readable text.

The command processor includes several features that make string processing easier. For example, you can use the INPUT 293, MSGBOX 313, and QUERYBOX 332 commands for user input; the ECHO and ECHOERR 253, ECHOS and ECHOSERR 255, SCREEN 342, SCRPUT 344, and VSCRPUT 391 commands for output; and the FOR 267 command or the @FILEREAD 460 function to scan through the lines of a file. In addition, variable functions 424 offer a wide range of strings and character handling

433 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 358 command at the beginning of this batch file saves the environment and aliases. Then the UNALIAS * 386 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 293 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 501 function numbers the first word as 0, not as 1.

The IFF 287 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 256 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 36 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 166 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 354 command to temporarily disable redirection symbols and other special characters. The two-line batch file above would be a lot more likely to produce the expected results if it were rewritten this way:

```
setdos /x-15678
input Enter a string:  %%str
echo %str
setdos /x0
```

The first line turns off alias processing and disables several special symbols, including the command separator and all redirection symbols. Once the string has been processed, the last line re-enables the features that were turned off in the first line.

If you need advanced string processing capabilities beyond those provided by the command processor, you may want to consider using the Perl 175, REXX 175, or Ruby 175 languages. Our products can execute Perl, REXX, and Ruby programs internally, and also support evaluating individual Perl, Rexx, and Ruby expressions internally.

6.2.13 Batch File Line Continuation

The command processor will combine multiple lines in the batch file into a single line for processing when the Escape Character 69 (the actual token or the symbolic "%= 410" reference) is the last character of each line to be combined (except the last). For example:

```
c:\> echo The quick brown fox jumped over the ^
sleeping ^
dog. > alphabet
```

You cannot use this technique to extend a batch file line beyond the normal command line length limit 71.

6.2.14 Batch File Compression

You can compress your *.BTM* files with BATCOMP 200. That command compresses batch files by about a third and makes them unreadable with the LIST 298 command and similar utilities. Compressed batch files run at approximately the same speed as uncompressed *.BTM* files.

You may want to consider compressing batch files if you need to distribute them to others and keep your original code secret or prevent your users from altering them. You may also want to consider compressing batch files to save some disk space on the systems where compressed files are used.

The full syntax for the batch compression command is

```
BATCOMP [/Ekkkk /K][/Q][/O] InputFile [OutputFile]
```

You must specify the full name of the input file, including its extension, on the BATCOMP command line. If you do not specify the output file, BATCOMP will use the same base name as the input file and add a *.BTM* extension. For example, to compress *MYBATCH.CMD* and save the result as *MYBATCH.BTM*, you can use either one of these commands:

```
batcomp mybatch.cmd
batcomp mybatch.cmd mybatch.btm
```

If the output file (*MYBATCH.BTM* in the examples above) already exists, BATCOMP will prompt you before overwriting the file. You can disable the prompt by including **/O** on the BATCOMP command line immediately before the input file name. Even if you use the /O option, BATCOMP will not compress a file into itself.

By default, BATCOMP does not remove comment lines, i.e. lines starting with **REM** or **::**, since in some instances, such as when a comment line occurs between a TEXT 376 and ENDTEXT, the compressed file would behave differently from the original by not displaying the comment line. To override that default and force deletion of comment lines, use the **/K** option. Lines starting with "REM >" (used to create a new, empty file, or to delete the old contents of one) are never considered comments to be removed.

The **/Q** ("quiet") option suppresses informational messages from BATCOMP.

JP Software does not provide a utility to decompress batch files. If you use BATCOMP, make sure that you also keep a copy of the original batch file for future inspection or modification.

You can adopt one of two strategies for keeping track of your original source files and compressed batch files. First, you may want to create the source files with a *.BAT* or *.CMD* extension and reserve the *.BTM* extension for compressed batch files. The advantage of this approach is that you can modify and test the uncompressed versions at any time, although they will run in the slower, traditional mode unless they begin with a LOADBTM 303 command.

If you prefer, you can use a *.BTM* extension for both the source and compressed files. In this case you will have to use a different directory or a different base name for each file. For example, you might use *SOURCE\MYBATCH.BTM* for the source file and *COMP\MYBATCH.BTM* for the compressed version, or use *MYBATCHS.BTM* for the source file and *MYBATCH.BTM* for the compressed file (however, the latter approach may make it more difficult to keep track of the correspondence between the source file and the compressed file).

If you plan to distribute batch files to users of different platforms, see Special Character Compatibility 72 for important information on the command separator, escape character, and parameter character used in each product.

The BATCOMP 200 command replaces the external program BATCOM32.EXE which was included in previous versions and is now obsolete.

6.2.15 REXX Support

REXX is a powerful file and text processing language developed by IBM, and available on many platforms. REXX is an ideal extension to the command processor batch language, especially if you need advanced string processing capabilities.

The REXX language is not built into the command processor, and must be obtained separately through (free) add-on REXX software, such as Open Object REXX (http://www.oorexx.org/) or Regina REXX (http://regina-rexx.sourceforge.net/).

4NT and ***TC*** automatically recognize the presence of a REXX interpreter on your system. When the command processor loads, it asks Windows to locate specific REXX libraries associated with Open Object REXX or Regina REXX. Specifically, it looks for *REGINA.DLL*, *WREXX32.DLL*, *RXREXX.DLL*, *REXX.DLL*, or *REXXAPI.DLL*. If a suitable library is found, the command processor checks to see if you are running a ***.REX*** or ***.REXX*** file, or if the first two characters on the first line of a ***.CMD*** file are [***/****], the beginning of a REXX comment. If either of these tests succeeds, the command processor passes the file to your REXX interpreter for processing.

(TC) When working with a REXX processor, ***TC*** automatically handles all input and output for the REXX program, and any standard REXX processor window for input and output is not displayed. If you need to run a REXX program inside your REXX processor's window, and not under ***TC***, you should start the REXX processor's executable file explicitly, then load and run the REXX program from there.

When you send a command from a REXX program back to the command processor to be executed (for example, if you execute a DIR command within a REXX script), the REXX software must use the correct address for the command processor. The command processor uses the address **CMD**.

For details on communication between REXX and the command processor, or for more information on any aspect of REXX, see your REXX documentation.

See also: the @REXX 487, @PERL 483 and @RUBY 488 functions.

6.2.16 Perl support

Perl is a powerful file and text processing language available on many platforms. Perl is an ideal extension to the command processor batch language, especially if you need advanced string processing capabilities.

The Perl language is not built into the command processor, and must be obtained separately. The version supported by ***4NT*** and ***TC*** is Active State Perl 5.8 (free from www.activestate.com).

4NT and ***TC*** automatically recognize the presence of a Perl interpreter on your system. If a suitable library is found, the command processor checks to see if you are running a ***.PL*** file. If so, the command processor passes the file to your Perl interpreter for processing.

See also: the @PERL 483, @REXX 487 and @RUBY 488 functions.

6.2.17 Ruby support

Ruby is a powerful object-oriented file and text processing language available on many platforms. Ruby is an ideal extension to the command processor batch language, especially if you need advanced string processing capabilities.

The Ruby language is not built into the command processor, and must be obtained separately. The version supported by ***4NT*** and ***TC*** is Ruby 1.8 (free from www.ruby-lang.org).

4NT and ***TC*** automatically recognize the presence of a Ruby interpreter on your system. If a suitable library is found, the command processor checks to see if you are running a ***.rb*** file. If so, the command processor passes the file to your Ruby interpreter for processing.

See also: the @RUBY 488, @REXX 487 and @PERL 483 functions.

6.2.18 EXTPROC and SHEBANG Support

For compatibility with CMD.EXE, the command processor 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. The command processor 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
```

The command processor would read the *GETDATA.CMD* file, determine that it began with an EXTPROC command, read the name of the processor program, and then execute the command:

```
D:\DATAACQ\DATALOAD.EXE D:\DATAACQ\GETDATA.CMD /p17
```

The hypothetical *DATALOAD.EXE* program would then be responsible for reopening the *GETDATA.CMD* file, ignoring the EXTPROC line at the start, and interpreting the other instructions in the file. It would also have to respond appropriately to the command line parameter entered (/p17).

Do not try to use the command processor as the external processor named on the EXTPROC line in the *.CMD* file. It will interpret the EXTPROC line as a command to reopen itself. The result will be an infinite loop that will continue until the computer runs out of resources and locks up.

4NT and ***TC*** also provide **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*.

7 Internal Commands

Our command processors give you instant access to more than 140 internal commands. (By contrast, Microsoft's CMD.EXE 513 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 177
- Commands By Category 181

Note: Remember that you can replace any internal command with an ALIAS 187 or plugin 325, or

disable an internal command with SETDOS /I 354.

7.1 Commands by Name

See also: Internal Commands Listed by Category 181

	Description	4NT	TC
? 185	Display list of internal commands or Prompt to execute a command	X	X
ACTIVATE 186	Activate or set window state	X	X
ALIAS 187	Define or display aliases	X	X
ASSOC 197	Windows file associations	X	X
ATTRIB 198	Change or display file attributes	X	X
BATCOMP 200	Batch file compression	X	X
BDEBUGGER 201	Batch file debugger	X	X
BEEP 208	Beep the speaker	X	X
BREAK 209	Define or display Ctrl-C state	X	X
BREAKPOINT 209	Set a batch debugger breakpoint	X	X
CALL 209	Call another batch file	X	X
CANCEL 211	End batch file processing	X	X
CD 211	Display or change directory	X	X
CDD 213	Change drive and directory	X	X
CHCP 215	Display or change code page	X	
CHDIR 211	Display or change directory	X	X
CLS 215	Clear the display window	X	X
COLOR 216	Change the display colors	X	X
COPY 216	Copy files and/or directories	X	X
DATE 224	Display or change date	X	X
DDEEXEC 224	Send DDE command		X
DEBUGSTRING 225	Send text to system debugger	X	X
DEL 225	Delete files and/or directories	X	X
DELAY 229	Wait for specified time	X	X
DESCRIBE 230	Display or change descriptions	X	X
DETACH 232	Start app detached	X	X
DIR 233	Display files and/or directories	X	X

DIRHISTORY 244	Display directory history list	X	X
DIRS 245	Display directory stack	X	X
DO 246	Create batch file loops	X	X
DRAWBOX 250	Draw a box	X	X
DRAWHLINE 251	Draw a horizontal line	X	X
DRAWVLINE 252	Draw a vertical line	X	X
ECHO 253	Echo a message	X	X
ECHOERR 253	Echo a message to stderr 571	X	X
ECHOS 255	Echo a message with no CR/LF	X	X
ECHOSERR 255	Echo with no CR/LF to stderr 571	X	X
EJECTMEDIA 256	Eject a removable drive	X	X
ENDLOCAL 256	Restore from a SETLOCAL	X	X
ERASE 225	Delete files and/or directories	X	X
ESET 258	Edit variables or aliases	X	X
EVENTLOG 259	Write Windows event log	X	X
EXCEPT 260	Exclude files from a command	X	X
EXIT 262	Exit the command processor	X	X
FFIND 262	Search for files or text	X	X
FOR 267	Repeat a command	X	X
FREE 274	Display disk space	X	X
FTYPE 274	Display or edit file types	X	X
FUNCTION 275	Create or edit user functions	X	X
GLOBAL 279	Run command in subdirectories	X	X
GOSUB 280	Call batch subroutines	X	X
GOTO 282	Branch in a batch file	X	X
HEAD 283	Display beginning of file	X	X
HELP 284	Help for internal commands	X	X
HISTORY 285	Display or change history	X	X
IF 286	Conditional command execution	X	X
IFF 287	Conditional command execution	X	X
IFTP 288	Open FTP connection	X	X
INKEY 291	Get a single keystroke	X	X
INPUT 293	Get a text string	X	X
JABBER 294	Send an IM	X	X

KEYBD 295	Set keyboard toggles	X	X
KEYS 295	Enable or disable history list	X	X
KEYSTACK 296	Send keystrokes to app	X	X
LIST 298	Display content of files	X	X
LOADBTM 303	Load batch file as .BTM	X	X
LOG 303	Save log of commands	X	X
MD 305	Create subdirectories	X	X
MEMORY 306	Display memory statistics	X	X
MKDIR 305	Create subdirectories	X	X
MKLNK 307	Create NTFS hard or soft link	X	X
MOVE 308	Move files or directories	X	X
MSGBOX 313	Popup message box	X	X
ON 315	Batch file error trapping	X	X
OPTION 317	Configure command processor	X	X
OSD 318	Display floating text	X	X
PATH 319	Set or display PATH	X	X
PAUSE 320	Wait for input	X	X
PDIR 320	User-formatted DIR	X	X
PLAYAVI 324	Display an .AVI file	X	X
PLAYSOUND 325	Play a sound file	X	X
PLUGIN 325	Load or unload plugin DLL	X	X
POPD 326	Restore from directory stack	X	X
POSTMSG 327	Send a message to a Window	X	X
PRINT 328	Print a file	X	X
PRIORITY 328	Set process priority	X	X
PROMPT 329	Change command line prompt	X	X
PUSHD 331	Save directory to stack	X	X
QUERYBOX 332	Popup input box	X	X
QUIT 333	Exit batch file	X	X
RD 333	Remove subdirectory	X	X
REBOOT 335	Reboot system	X	X
RECYCLE 336	Display or empty recycle bin	X	X
REXEC 340	Remotely execute commands	X	X
REM 336	Remark	X	X
REN 337	Rename files or directories	X	X
RENAME 337	Rename files or directories	X	X
RETURN 340	Return from GOSUB	X	X
RMDIR 333	Remove subdirectory	X	X
RSHELL 341	Remotely execute commands	X	X

SCREEN 342	Position cursor	X	X
SCRPUT 344	Write directly to screen	X	X
SELECT 345	Select files for a command	X	X
SENDMAIL 349	Send email	X	X
SET 351	Set or display environment variables	X	X
SETDOS 354	Set or display command processor options	X	X
SETLOCAL 358	Save environment, aliases and functions	X	X
SHIFT 359	Shift batch file parameters	X	X
SHORTCUT 360	Create a Windows shortcut	X	X
SHRALIAS 361	Share aliases	X	X
SMPP 362	Simple message transfer	X	X
SNMP 363	Send SNMP traps	X	X
SNPP 363	Send message to pager	X	X
START 364	Start a new session	X	X
SWITCH 367	Batch file switch / case	X	X
SYNC 369	Synchronize directories	X	X
TAIL 371	Display end of file	X	X
TASKEND 373	End a task	X	X
TASKLIST 374	Display Windows task list	X	X
TCTOOLBAR 374	Edit Toolbar		X
TEE 376	Pipe "tee-fitting"	X	X
TEXT 376	Display text in batch file	X	X
TIME 378	Set or display time	X	X
TIMER 379	Stopwatch	X	X
TITLE 380	Set window title	X	X
TOUCH 381	Change file timestamps	X	X
TRANSIENT 382	Toggle shell transient mode	X	X
TREE 383	Display directory tree	X	X
TRUENAME 384	Display true pathname	X	X
TYPE 384	Display files	X	X
UNALIAS 386	Remove aliases	X	X
UNFUNCTION 387	Remove user-defined functions	X	X
UNSET 388	Remove environment variable	X	X
VER 389	Display version	X	X
VERIFY 390	Display or set disk verification	X	X
VOL 390	Display or set disk volume label	X	X
VSCRPUT 391	Write text vertically	X	X

WHICH 391	Display command information	X	X
WINDOW 392	Window management	X	X
WMIQUERY	WMI queries	X	X
Y 395	Pipe "y-fitting"	X	X
	Description	4NT	TC

7.2 Commands by Category

See also: Internal Commands Listed by Name 177

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

		4NT	TC
ATTRIB 198	Change or display file attributes	X	X
COPY 216	Copy files and/or directories	X	X
DEL 225	Delete files and/or directories	X	X
DESCRIBE 230	Display or change descriptions	X	X
ERASE 225	Delete files and/or directories	X	X
FFIND 262	Search for files or text	X	X
HEAD 283	Display beginning of file	X	X
IFTP 288	Open FTP connection	X	X
LIST 298	Display contents of files	X	X
MOVE 308	Move files or subdirectories	X	X
RECYCLE 336	Display or empty recycle bin	X	X
REN 337	Rename files or directories	X	X
RENAME 337	Rename files or directories	X	X
SELECT 345	Select files for a command	X	X
SYNC 369	Synchronize directories	X	X
TAIL 371	Display end of file	X	X
TOUCH 381	Change file dates/times	X	X
TREE 383	Display directory tree	X	X

TRUENAME 384	Display true pathname	X	X
TYPE 384	Display files	X	X
Y 395	Pipe "y-fitting"	X	X

Subdirectory management

		4NT	TC
CD 211	Display or change directory	X	X
CDD 213	Change drive and directory	X	X
CHDIR 211	Display or change directory	X	X
DIR 233	Display files and/or directories	X	X
DIRS 245	Display directory stack	X	X
MD 305	Create subdirectories	X	X
MKDIR 305	Create subdirectories	X	X
MKLNK 307	Create NTFS hard or soft link	X	X
PDIR 320	User-formatted DIR	X	X
POPD 326	Restore from directory stack	X	X
PUSHD 331	Save directory to stack	X	X
RD 333	Remove subdirectory	X	X
RMDIR 333	Remove subdirectory	X	X

Input and output

		4NT	TC
DRAWBOX 250	Draw a box	X	X
DRAWHLINE 251	Draw a horizontal line	X	X
DRAWVLINE 252	Draw a vertical line	X	X
ECHO 253	Echo a message	X	X
ECHOERR 253	Echo a message to stderr	X	X
ECHOS 255	Echo a message with no CR/LF	X	X
ECHOSERR 255	Echo with no CR/LF to stderr	X	X
INKEY 291	Get a keystroke	X	X
INPUT 293	Get an input line	X	X
KEYSTACK 296	Send keystrokes to app	X	X
MSGBOX 313	Popup message box	X	X
OSD 318	Display floating text	X	X
PLAYAVI 324	Play an .AVI file	X	X
PLAYSOUND 325	Play a sound file	X	X
PRINT 328	Print a file	X	X
QUERYBOX 332	Popup input box	X	X

SCREEN 342	Position cursor	X	X
SCRPUT 344	Write directly to screen	X	X
SENDMAIL 349	Send email	X	X
SMPP 362	Send SMS message	X	X
SNMP 363	Send SNMP trap	X	X
SNPP 363	Send message to pager	X	X
VSCRPUT 391	Write text vertically	X	X

Window management commands

		4NT	TC
ACTIVATE 186	Activate or set window state	X	X
POSTMSG 327	Send a message to a Window	X	X
TITLE 380	Set window title	X	X
WINDOW 392	Window management	X	X

Commands primarily for use in or with batch files and aliases
(some work only in batch files; see the individual commands for details)

		4NT	TC
ALIAS 187	Define or display aliases	X	X
BATCOMP 200	Batch file compression	X	X
BDEBUGGER 201	Batch file debugger	X	X
BEEP 208	Beep the speaker	X	X
BREAKPOINT 209	Set a batch debugger breakpoint	X	X
CALL 209	Call another batch file	X	X
CANCEL 211	End batch file processing	X	X
DDEEXEC 224	Send DDE command		X
DEBUGSTRING 225	Send text to system debugger	X	X
DELAY 229	Wait for specified time	X	X
DO 246	Batch file looping	X	X
ENDLOCAL 256	Restore a SETLOCAL	X	X
EJECTMEDIA 256	Eject a removable drive	X	X
FOR 267	Repeat a command	X	X
FUNCTION 275	Create or edit user functions	X	X
GLOBAL 279	Run command in subdirectories	X	X
GOSUB 280	Call batch subroutines	X	X
GOTO 282	Go to a batch file label	X	X
IF 286	Conditional command execution	X	X
IFF 287	Conditional command execution	X	X
JABBER 294	Send an IM	X	X
LOADBTM 303	Load batch files as .BTM	X	X
ON 315	Batch file error trapping	X	X
PAUSE 320	Wait for input	X	X

QUIT 333	Exit batch file	X	X
REM 336	Remark	X	X
RETURN 340	Return from GOSUB	X	X
SETLOCAL 358	Save environment, aliases, and functions	X	X
SHIFT 359	Shift batch file parameters	X	X
SWITCH 367	Batch file switch / case	X	X
TEXT 376	Display text in batch file	X	X
TRANSIENT 382	Toggle shell transient mode	X	X
UNALIAS 386	Remove aliases	X	X
UNFUNCTION 387	Remove user-defined functions	X	X

Environment and path commands

		4NT	TC
ESET 258	Edit variables or aliases	X	X
PATH 319	Set or display PATH	X	X
SET 351	Set or display environment variables	X	X
UNSET 388	Remove environment variables	X	X

System configuration and status

		4NT	TC
ASSOC 197	Windows file associations	X	X
BREAK 209	Define or display Ctrl-C state	X	X
CHCP 215	Display or change code page	X	X
CLS 215	Clear the display window	X	X
COLOR 216	Change the display colors	X	X
DATE 224	Display or change date	X	X
DIRHISTORY 244	Display directory history list	X	X
EVENTLOG 259	Write to Windows event log	X	X
FREE 274	Display disk space	X	X
FTYPE 274	Display or edit file types	X	X
HISTORY 285	Display or change history	X	X
KEYBD 295	Set keyboard toggles	X	X
KEYS 295	Enable or disable history list	X	X
LOG 303	Save log of commands	X	X
MEMORY 306	Display memory statistics	X	X
OPTION 317	Configure command processor	X	X
PLUGIN 325	Load or unload plugin DLL	X	X
PROMPT 329	Change command line prompt	X	X

REBOOT 335	Reboot system	X	X
SETDOS 354	Command processor options	X	X
SHORTCUT 360	Create a Windows shortcut	X	X
TASKEND 373	End a task	X	X
TASKLIST 374	Display Windows task list	X	X
TCTOOLBAR 374	Edit TC toolbar		X
TIME 378	Set or display time	X	X
VERIFY 390	Display or set disk verification	X	X
VER 389	Display version	X	X
VOL 390	Display or set disk volume label	X	X

Other commands

		4NT	TC
? 185	Display list of internal commands, or prompt to execute a command	X	X
DETACH 232	Start app detached	X	X
EXCEPT 260	Exclude files from a command	X	X
EXIT 262	Exit the command processor	X	X
HELP 284	Command processor help	X	X
SHRALIAS 361	Share aliases & functions	X	X
START 364	Start a new session	X	X
REXEC 340	Remotely execute command	X	X
RSHELL 341	Remotely execute command	X	X
TEE 376	Pipe "tee-fitting"	X	X
TIMER 379	Stopwatch	X	X
WHICH 391	Display command information	X	X

7.3 ? (command)

Purpose: Display a list of internal and plugin commands, or prompt for a command.

Format: ? ["prompt" command]

Usage:

The **?** command has two separate meanings:

1. When you use the ? command by itself, it displays a list of internal and plugin commands. For help with any individual command, see the HELP 284 command. If you have disabled a command with SETDOS /I 354, 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 209 command will be executed:

7.4 ACTIVATE

Purpose: Activate a window, set its state, or change its title.

Format: ACTIVATE [/R] "title" [MAX | MIN | RESTORE | CLOSE | TOPMOST | NOTOPMOST | TOP | BOTTOM | HIDE | /POS=left,top,width,height | /TRANS=n | TRAY | "newtitle"]

title	Current title of the window to be activated
left	New location of the left border of the window, in pixels
top	New location of the top border of the window, in pixels
width	New width of the window, in pixels
height	New height of the window, in pixels
newtitle	New title for window

/R(estore original window)

See also: START 364, TITLE 380, and WINDOW 392.

Usage:

ACTIVATE 186 activates, and optionally modifies, another session's window. It is not intended to modify the characteristics of the current command processor session (use TITLE 380 or WINDOW 392 for that purpose).

Title specifies the name of the target window to be activated. You can use wildcards 19, including extended wildcards, in ***title***. This is useful with applications that change their window title to reflect the file currently in use. ***Title*** must be enclosed in quotes.

Each execution of ACTIVATE allows you to modify one property of the target window. To perform multiple operations, use multiple ACTIVATE commands.

The options are:

MAX	Expands the window to its maximum size and activates it.
MIN	Reduces the window to an icon.
RESTORE	Activates the window at its default size and location.
CLOSE	Sends a "close" message to close the window.
POS	Sets the window position and size (in pixels).
TOPMOST	Keeps the window on top of all other windows until it closes, or **NOTOPMOST** is used.
TRANS	Transparency level, where n=0 (invisible) to 255 (opaque) (does not work for console windows)

NOTOPMOST	Allows other windows to overlay the window (this is the normal state for most windows).
TOP	Moves the window to the top of the window order, above all other non-**TOPMOST** windows.
BOTTOM	Moves the window to the bottom of the window order.
HIDE	Makes the window invisible (to make the window visible again, use **RESTORE**).
TRAY	Move the specified window to the system tray.
"newtitle"	Changes the window title.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

If you specify ***newtitle***, it must be enclosed in double quotes (which will not appear as part of the title text).

ACTIVATE is often used before KEYSTACK 296 to make sure the proper window receives the keystrokes.

ACTIVATE works by sending messages to the named ***window***. If the window ignores or misinterprets the messages, ACTIVATE may not have the effect you want.

If ACTIVATE is used in a batch file, and the batch file is not itself running in the active window (the window with its title bar highlighted), then ACTIVATE may not activate the desired window. This is because under Windows you cannot make another window active except when the window which issues the command is itself active already. This is a Windows feature which helps to prevent windows which are not in the foreground from grabbing input intended for other windows.

Example

The example below first maximizes, and then renames the window originally called "Take Command":

```
activate "Take Command" max
activate "Take Command" "Take Command is Great!"
```

7.5 ALIAS

Purpose: Create new command names that execute one or more commands or redefine default options for existing commands; assign commands to keystrokes; load or display the list of defined alias names.

Format: Display mode:
ALIAS [/G /L /P] [wildname]

Definition mode:
ALIAS /R [file...] | name[=]value

file	One or more input files to read alias definitions from
wildname	Name of alias whose definition is to be displayed (may contain ***** and **?** wildcards)
name	Name for an alias, or for the key to execute the alias
value	Text to be substituted for the alias name or key

/G(lobal) 196 /P(ause) 196

/L(ocal) 196 /R(ead file) 196

See also: UNALIAS 386, ESET 258, and Aliases 160.

Usage:

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 233 /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.

(TC) You can also define or modify aliases with the Alias Window 81. All of the information in this section also applies to aliases defined via the dialog, unless otherwise noted.

If you define aliases for commonly used application programs, you can often remove the directories they're stored in from the PATH 399. 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 the command processor had to search the PATH 399 for it. In addition, PATH 399 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 399 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 399, you will need to

define aliases for all the executable programs you commonly use which are stored in that directory.

4NT and ***TC*** also support Directory Aliases 18, a shorthand way of specifying pathnames.

Aliases are stored in memory, and are not saved automatically when you turn off your computer or end your current command processor 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 191 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 211 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 65, and the parameter character 135, and may also use the escape character 69.

When an alias contains multiple commands, the commands are executed one after the other. However, if any of the commands runs an external Windows application, you must be sure the alias will wait for the application to finish before continuing with the other commands. This behavior is controlled by the **Wait for completion** setting in the configuration dialogs 151 or the ExecWait 118 directive in the INI file 91.

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 286 or IFF 287 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 331 command, runs the ***WP*** alias, and then returns to the original directory with the POPD 326 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 the command processor not to attempt to interpret that command as another (nested) alias. An asterisk used this way must be preceded by a space or the command separator and followed immediately by an internal or external command name.

By using an asterisk, you can redefine the default options for any internal or external command. For example, suppose that you always want to use the DIR 233 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 the command processor would repeatedly re invoke the **DIR** alias, rather than running the DIR 233 command. This would cause an "Alias loop" or "Command line too long" error. The asterisk forces interpretation of the second DIR 233 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 230 command will run anytime you type DESCRIBE. But two simple aliases will give you access to both the *DESCRIBE.EXE* program and the DESCRIBE 230 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 230 command. The second line defines **FILEDESC** as a new name for the internal DESCRIBE 230 command. The asterisk is needed in the second command to indicate that the following word means the internal command DESCRIBE 230, not the **describe** alias which runs your external program.

Another way to understand the asterisk is to remember that a command is always checked for an alias first, then for an internal or external command, or a batch file. The asterisk at the beginning of a command name simply skips over the usual check for aliases when processing that command, and allows the command processor to go straight to checking for an internal command, external command, or batch file.

You can 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 233 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 354 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 191 and autoexecute 192.

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, the command processor 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, the command processor 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 213 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 117 directive (by default, the **Esc** key, best represented as **%=e**) as the first character of the alias value.

To force a visible indication that an autoexecute keystroke alias was used, include a descriptive ECHO 253 command in the alias value.

Be careful not to assign aliases to keys that are already used at the command line (e.g., ***F1*** for HELP 284). The command line meanings take precedence and the keystroke alias will never be invoked. If you want to use one of the command line keys for an alias instead of its normal meaning, you must first disable its default use with the NormalKey 134 or NormalEditKey 134 directives in the INI file 91.

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 550 for a complete listing of key names and a description of the key name format. You can also define a keystroke alias by using @ or @@ plus a scan code for one of the permissible keys (see the Key Code and Scan Code Tables 545 for a list of scan codes). In most cases it will be easier to use key names. Scan codes should only be used with unusual keyboards where a key name is not available for the key you are using.

(TC) Windows always interprets **Alt** plus a key as a menu accelerator key. Such keystrokes are not passed to ***TC***, and therefore cannot be used for keystroke aliases.

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 the command processor starts, you can edit them with a text editor, reload the edited file with **ALIAS /R**, and know that the same alias list will be loaded the next time you start the command processor.

When you define aliases in a file that will be read with the **ALIAS /R** command, you do not need back quotes around the value, even if back quotes would normally be required when defining the same alias at the command line or in a batch file.

(TC) You can also save and reload your aliases using the Aliases dialog 81. The Export button in the dialog box is equivalent to the **ALIAS** > ***filename*** command shown above, and the Import button is equivalent to the **ALIAS /R** command.

To remove an alias, use the UNALIAS 386 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 166 for details. Alias examples in this section assume the ***4NT*** and ***TC*** default of ParameterChar=$. If you are using a different ParameterChar 135 you will need to edit the examples accordingly.

Parameters that are referred to in an alias, but which are missing on the command line, appear as empty strings inside the alias. For example, if you only put two parameters on the command line, any reference in the alias to **%3** or any higher-numbered parameter will be interpreted as an empty string.

The parameter **%n$** has a special meaning. The command processor interprets it to mean "the entire command line, from parameter ***n*** to the end." If ***n*** is not specified, it has a default value of **1**, so **%$** 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 287 command to redefine the defaults for SET 351. It should be entered on one line:

```
alias set = `iff %# == 0 then & *set /p & else & *set %$ & endiff`
```

This modifies the SET 351 command so that if SET 351 is entered with no parameters, it is replaced by SET /P (pause after displaying each page), but if SET 351 is followed by a parameter, it behaves normally. Note the use of asterisks (*set) to prevent alias loops.

If an alias uses parameters, command line parameters will be deleted up to and including the highest referenced parameter. For example, if an alias refers only to **%1** and **%4**, then the first and fourth parameters will be used, the second and third parameters will be discarded, and any additional parameters beyond the fourth will be appended to the expanded command (after the ***value*** portion of the alias). If an alias uses no parameters, all of the command line parameters will be appended to the expanded command. A convenient way to prevent unwanted command line parameters from being appended is to add a reference to **%511** within the alias.

Aliases also have full access to all variables in the environment, internal variables, and variable functions. For example, you can create a simple command line calculator this way:

```
alias calc = `echo The answer is: %@eval[%$]`
```

Now, if you enter:

```
calc 5 * 6
```

The alias will display:

```
The answer is: 30
```

Expanding Aliases at the Prompt

You can expand an alias on the command line and view or edit the results by pressing **Ctrl-F** after typing the alias name, but before the command is executed. This replaces the alias with its contents, and substitutes values for each alias parameter, just as if you had pressed the **Enter** key. However, the command is not executed; it is simply redisplayed on the command line for additional editing.

Ctrl-F is especially useful when you are developing and debugging a complex alias, or if you want to make sure that an alias that you may have forgotten won't change the effect of your command.

Local and Global Aliases

Aliases can be stored in either a local or global list. The selection is made during command processor startup, using the **/L** or **/LA** START 364 or startup options 4, or by the LocalAliases 130 directive, or interactively with the **ALIAS /G** and **ALIAS /L** options. The global alias list is limited to 128K characters; the local alias list is limited only by memory size.

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

With a global alias list, all copies of the command processor, 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 ***4NT*** and ***TC***.

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

When you use SETLOCAL 358 / ENDLOCAL 256 inside a batch file, changes in alias definitions are restored by the ENDLOCAL 256. However, if the session uses the global alias list, any concurrent sessions also using the global alias list are affected.

Whenever you start a second copy of the command processor which uses a local alias list, it inherits a copy of the aliases from the previous shell. However, any changes to the aliases made in the secondary shell will affect only that shell. If you want changes made in a secondary shell to affect the previous shell, use a global alias list in both shells.

Retaining Global Aliases with SHRALIAS

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

If you want the alias list to be retained in memory even when no command processor session is running, you need to execute the SHRALIAS 361 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 361 from your TCSTART / 4START 8 file.

SHRALIAS 361 retains the alias list in memory, but cannot preserve it when Windows itself is shut down. To save your aliases when restarting Windows, you must store them in a file and reload them after the system restarts. For details on how to do so, see Saving and Reloading Your Aliases 192 above.

The PRE_INPUT, PRE_EXEC, and POST_EXEC Aliases

When at the command prompt (i.e., not executing a batch file), the command processor will look for (and execute them if found) the following aliases:

PRE_INPUT - executed immediately before accepting input for a new command line.

PRE_EXEC - executed immediately after a command line is entered (before any expansion or redirection).

POST_EXEC - executed immediately after returning from a command and before displaying the prompt.

None of these aliases will be passed any arguments.

The UNKNOWN_CMD Alias

If you create an alias with the name **UNKNOWN_CMD**, it will be executed any time the command processor 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 "%$"`
```

(using the ***4NT***/***TC*** default of ParameterChar=$ 135).

If the **UNKNOWN_CMD** alias contains an unknown command, it will call itself repeatedly. If this occurs, the command processor will loop up to 10 times, then display the **UNKNOWN_CMD loop** error.

Warnings

When you define an alias in the command line (i.e., without using the /R 195 option), variables and functions not protected by back quotes or doubled % signs are immediately evaluated, and the result becomes part of the alias value.

Syntax errors in an alias are not detected until the alias is executed.

Options:

/G Switch from a local to a global alias list.

/L Switch from a global to a local alias list.

/P This option is only effective when ALIAS is used to display existing definitions. It pauses the display after each page and waits for a keystroke before continuing (see Page and File Prompts 41).

/R This option loads an alias list from a file. The format of the file is the same as that of the ALIAS display:

name=value

where ***name*** is the *name* of the alias and ***value*** is its *value*. You can use an equal sign **=** or space to separate ***name*** and ***value***. Back quotes are not required around the value. Variables and functions referenced in the definitions remain in the definitions, to be evaluated each time the alias is executed. You can add comments to the file by starting each comment line with a colon **:**. You can load multiple files with one **ALIAS /R** command by placing the names on the command line, separated by spaces:

```
alias /r alias1.lst alias2.lst
```

Each definition in an **ALIAS /R** file can be up to 8,191 characters long. The definitions can span multiple lines in the file if each line of the definition, except the last, is terminated with an escape character 69.

ALIAS /R will read from stdin 571 if no filename is specified and input is redirected:

```
alias /r <
```

7.6 ASSOC

Purpose: Modify or display relationships between file extensions and file types stored in the Windows registry.

Format: ASSOC [/P /R [*file*...] | [.*ext*[=[*filetype*]]]

file	One or more input files to read association definitions from.
.ext	The file extension whose file type you want to display or set.
filetype	A file type stored in the Windows registry.

/P(ause) 197 /R(ead) 197

See also: FTYPE 274, and Executable Extensions 33.

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 535. 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 274 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 41.

/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
```

ASSOC /R will read from stdin 571 if no filename is specified and input is redirected.

7.7 ATTRIB

Purpose: Change or view file and subdirectory attributes.

Format: ATTRIB [/A:[[-+]rhsa] /D /E /I"text" /P /Q /S[n]] [+|-[ADHORST]] [@*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 32 for details).

/A: (Attribute select) 199
/D(irectories) 199
/E (No error messages) 199
/I"text" (match description) 200
/P(ause) 200
/Q(uiet) 200
/S(ubdirectories) 200

Attribute flags:

Clear	Set	Attribute affected
-A	+A	archive
-H	+H	hidden
-O	+O	offline
-T	+T	temporary
-R	+R	read-only
-S	+S	system

File Selection

Supports attribute switches 28, extended wildcards 19, ranges 22, multiple file names 29, and include lists 30. Use wildcards with caution on LFN volumes; see LFN File Searches 31 for details.

Usage:

Every file and subdirectory has attributes that can be turned on (set) or turned off (cleared): **Archive**, **Hidden**, **Read-only**, **System, Normal**, **Offline**, and **Temporary**. For details on the meaning of each attribute, see File Attributes 524.

The ATTRIB command lets you view, set, or clear attributes for any file, group of files, or subdirectory.

You can view file attributes by entering ATTRIB without specifying new attributes (*i.e.*, without the **[+|-[ADHORST]]** part of the format), or with the DIR /T 233 command.

The primary use of ATTRIB is to set attributes. For example, you can set the read-only and hidden attributes for the file *MEMO*:

```
attrib +rh memo
```

Attribute options apply to the file(s) that follow the options on the ATTRIB command line. The example below shows how to set different attributes on different files with a single command. It sets the archive attribute for all *.TXT* files, then sets the system attribute and clears the archive attribute for *TEST.COM*:

```
attrib +a *.txt +s -a test.com
```

When you use ATTRIB on an LFN drive, you must double quote any file names which contain white

space or special characters.

To change directory attributes, use the **/D** switch. If you give ATTRIB a directory name instead of a file name, and omit **/D**, it will append "*.*" to the end of the name and act on all files in that directory, rather than acting on the directory itself.

NTFS also supports **D** (subdirectory), **V** (volume label), **E** (encrypted), **C** (compressed), **I** (not content-indexed), **J** (junction / reparse point) and **P** (sparse file) attributes. These attributes will be displayed by ATTRIB, but cannot be altered; they are designed to be controlled only by Windows.

ATTRIB will ignore underlines in the new attribute (the **[+|-[ADHORST]]** 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 439 variable function or from ATTRIB itself (both of which use underscores to represent attributes that are not set) and send that string back to ATTRIB to set attributes for other files. For example, to clear the attributes of *FILE2* and then set its attributes to match those of *FILE1*:

```
attrib -arhs file2 & attrib +%@attrib[file1] file2
```

When ATTRIB encounters a **+D** or **-D** in the attribute string it treats it as equivalent to the **/D** switch, and allows modification of the attributes of a directory. When combined with @ATTRIB, or with ATTRIB's output, both of which return a **D** to signify a directory, this feature allows you to transfer attributes from one directory to another. For example, to clear the attributes of all files and directories beginning with *ABC* and then set their attributes to match those of *FILE1* (enter this on one line):

```
attrib -arhs abc* & attrib +%@attrib[file1] abc*
```

Options:

/A: Select only those files that have the specified attribute(s) set. See Attribute Switches 28 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 32 for details.

/D If you use the **/D** option, ATTRIB will modify the attributes of directories in addition to files (yes, you can have a hidden directory):

```
attrib /d +h c:\mydir
```

If you use a directory name instead of a file name, and omit **/D**, ATTRIB will append "*.*" to the end of the name and act on all files in that directory, rather than acting on the directory itself.

/E Suppress all non-fatal error messages, such as "File Not Found." Fatal error messages, such as "Drive not ready," will still be displayed. This option is most useful in batch files and aliases, and when recursing through the directory hierarchy, where many directories

have no files matching your selection criteria. .

/I"text" Select files by matching text in their descriptions. The text can include wildcards 19 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 32 for details

/P Wait for a key to be pressed after each screen page before continuing the display. Your options at the prompt are explained in detail under Page and File Prompts 41

.

/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 32 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.

7.8 BATCOMP

Purpose: Compress and optionally encrypt batch files. See Batch File Compression 173 for additional details.

Format: BATCOMP [/Ekkkk /K /O /Q] InputFile [OutputFile]

Inputfile A file to compress and/or encrypt.
OutputFile A file to hold the output from the command

/E(ncryption key) 201 /O(verwrite) 201
/K (remove comments) 201 /Q(uiet) 201

File Selection

The single input file must be specified explicitly.

Usage:

BATCOMP is a batch file compressor and optionally allows for simple key-based encryption.

If you do not specify ***OutputFile***, it defaults to the ***InputFile*** name with a **BTM** extension. If you specify ***OutputFile*****,** you must give an extension or the output file will not have one. In other words there is no BTM default if you just give a file name. If ***OutputFile*** already exists it will not be overwritten unless **/O** is used.

The output BTM file will not be legible, but it will run under the current versions of ***4NT*** and ***TC***. The behavior and performance of the file should be the same as if it were run in its original source form as a ***.BTM*** file.

Compression is not effective for very small files and may even result in a slightly larger file.

Options:

/E Specify a key (string) to be used for encryption.

/K By defaults, comments (i.e. lines starting with **REM** or **::** but not **REM >**) are retained in the compressed file. /K forces those lines to be stripped for additional size reduction.

/O Forces overwriting of any existing ***OutputFile***.

/Q Suppresses all progress reports (stdout 571). Errors (stderr 571) are still shown.

7.9 BDEBUGGER

Purpose: Calls the internal batch file debugger. This is an extremely powerful command that includes support for breakpoints, bookmarks, syntax coloring, and editing.

Format: BDEBUGGER [/A /B /C /E /F /I /K /N /W] batchfilename [parameters]

batchfilename	Full name of the batch file to debug.
parameters	parameters for the batch file

/A(lias list window) 207
/B(atch variable window) 207
/C(reate new batch file) 207
/E(nvironment variable window) 207
/F (user-defined Functions window) 207
/I(nactive) 207
/K(eep debugger window) 207
/N (do Not hide the debugger window) 208
/W(atch window) 208

Usage:

BDEBUGGER opens a special window in which a batch file can be examined, modified, and executed.

The debugger window includes a slider control on the lower left corner of the status bar to control the transparency level of the debugger window. If the debugger window is at least partially transparent, it will not be hidden while individual commands in the batch file are being executed (see the /N option).

The batch debugger window toolbar has a number of icons to control debugging. Each has a tooltip for quick reference:

New	Create a new batch file. If you have an existing file open, you will be prompted to save any changes before it exits.
Open	Open an existing batch file.
Save	Save the current batch file.
Print	Print the current batch file.
Copy	Copy the highlighted selection to the clipboard.
Cut	Copy the highlighted selection to the clipboard and delete it from the file.
Paste	Copy the contents of the clipboard to the current cursor location.
Find	Search for text.
Replace	Replace the text with other text.
Undo	Undo the last edit.
Redo	Restore the last Undo.
Start Debugging	Starts the debugger. The cursor will be placed on the first line.

Stop Debugging	Stops the debugger.
Step Into	Execute the current line.
Step Over	Execute the current line but turn off the debugger for the duration of a CALL or GOSUB.
Run to Breakpoint	Execute the batch file, stopping at the next breakpoint.
Toggle Breakpoint	Sets or turns off a breakpoint on the current line.
Clear Breakpoints	Turns off all breakpoints.
File Properties	Displays information on the current batch file.
Start New Shell	Start another copy of the command processor (this is useful if you need to perform some tasks while debugging a file.)

You can also set a breakpoint by moving the mouse cursor to the left margin of a line and left-clicking. You can only set a breakpoint on an executable line (i.e., not on a blank line, comment, label, etc.),

You can get help for the currently selected (highlighted) command / variable / function by pressing Ctrl-F1, or right-clicking the mouse and selecting **Help** from the context menu.

You can change the line to be executed next when in debugging mode with either the "GOTO" dialog or by moving the caret to the line and either right clicking & selecting "Jump to This Line" or by pressing Ctrl-Shift-F11. Note that if you attempt to jump into or out of a DO loop or IFF block, bad things will happen!

The debugger's Watch window allows you to monitor environment variables or to pause execution when a specified condition is met. The Watch window appears at the bottom of the debugger window. Enter the variable name or expression in the left column; the debugger will automatically display the current value in the right column. You can also add a variable to the Watch window by selecting it in the main debugger window, then clicking the right mouse button and selecting "Add to Watch". If the string in the left column is a single argument, it is assumed to be a variable name. Otherwise, it is assumed to be an expression. Expressions can be anything that IF can evaluate; for example:

```
%i = 3
ERR0RLEVEL GT 12
```

Note that expressions require variable names to be prefixed by a %. If you're entering a single variable argument to monitor, do not use a %.

When the value of a monitored variable changes, the Watch window will change its background color (to a light red).

Ctrl-F9 will invoke the Evaluate Expression dialog for the current selection, or if no text is selected, for the current line.

You can open popup windows to display and/or modify aliases, batch variables, environment variables, and user functions. These windows will "dock" to the outer edges of the debugger window and will move with the debugger window. The variable windows also have a toolbar, with a couple of additional buttons:

Update List	Save a modified list.
Import a File	Add the contents of a file to the list.

The environment variables window displays any variables changed during debugging in a different color. (The default is red; you can change it by adding a ChangeRGB=... line to BATCH.BCP with the appropriate RGB value.) You can specify variables to exclude from the environment variable window with the DebugVariableExclude variable. For example, to suppress the display of the

processor and user variables:

```
set DebugVariableExclude=proc*;user*
```

Note that this option doesn't affect the existence of the variables, just whether they're displayed in the environment variable window.

If you press **Ctrl-C** or **Ctrl-Break** while debugging, you will see the prompt:

Cancel batch job ***filename*** (Y/N/A/D) :

Pressing **D** will return you to single-step mode in the debugger. (This allows you to interrupt a **run-to-breakpoint** without terminating the debugger and batch file.)

The default settings for the BDEBUGGER (Batch File Debugger) command are kept in ASCII text file BATCH.BCP (in the program directory). Only modify the contents of that file if you're positive you know what you're doing. We strongly recommend that you keep a copy of the original, should your modifications prove unworkable or undesirable.

The text processing commands available in the batch debugger and variable windows are listed below. The default "hard coded" values used in the BATCH.BCP file are shown in parentheses after each command name. The text commands can be classified into general categories:

- **Caret commands**

Right	(1)	This command will move the caret one character to the right. When the caret is on the last position of the current line it is moved to the first position of the next line.
Shift-Right	(2)	In addition to the caret movement this command will also extend the current selection to the new caret position.
Left	(3)	This command will move the caret one character to the left. When the caret is on the first position of the current line it is moved to the last position of the previous line.
Shift-Left	(4)	In addition to the caret movement, this will also extend the current selection to the new position.
Up	(5)	This command will move the caret one line up. The caret column position will be set as close to its previous column position as possible.
Shift-Up	(6)	In addition to the caret movement this command will also extend the current selection to the new position.
Down	(7)	This command will move the caret one line down. The caret column position will be set as close to 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	(8)	In addition to the caret movement this command will also extend the current selection to the new position.

End (9) This command will move the caret to the end of the line it is currently on. If the caret is already at the end nothing happens.

Shift-End (10) In addition to the caret movement this command will also extend the current selection to the new position.

Home (11) This command will move the caret to the start of the line it is currently on. If the caret is already at the start nothing happens.

Shift-Home (12) In addition to the caret movement this command will also extend the current selection to the new position.

Ctrl-Right (13) This command will move in one of the following ways:

- When the caret is located on a delimiter character the caret is moved right until the first non-delimiter and non-white space is found.
- When the caret is located on a non-delimiter character and not on a white space character the caret is moved to the start of the next word.
- When the caret is located on a white space the caret is moved to the start of the next word or the next delimiter depending on what comes first.
- When the caret is located on the last word, delimiters or white-space of the current line the caret is moved to the first word or delimiter of the next line.

Ctrl-Shift-Right (14) In addition to the caret movement this command will also extend the current selection to the new caret position.

Ctrl-Left (15) This command will move in one of the following ways:

- When the caret is located on a delimiter character and it is preceded by delimiters the caret is moved left to the first delimiter.
- When the caret is located on a delimiter character and it is not preceded by delimiters the caret is move to the start of the previous word or delimiters.
- When the caret is located on a non-delimiter character and not on a white-space character the caret is moved to the start of the current word.
- When the caret is located on a white space the caret is moved to the start of the previous word or the previous delimiters depending on what comes first.
- When the caret is located on the start of the first word, delimiters or white-space of the current line the caret is moved to the start of the last word or delimiters of the previous line.

Ctrl-Shift-Left (16) In addition to the caret movement this command will also extend the current selection to the new position.

Ctrl-Home (17) This command will move the caret to the beginning of the text. When the caret is already at this location nothing happens.

Ctrl-Shift-Home (18) In addition to the caret movement this command will also extend the current selection to the new position.

Ctrl-End (19) This command will move the caret to the end of the text. When the caret is already at this location nothing happens.

Ctrl-Shift-End (20) In addition to the caret movement this command will also extend the current selection to the new position.

Ctrl-Tab (21) This command will move the caret to the previous tab-stop position. When the caret is located at the start of the line nothing happens.

Ctrl-Shift-Tab (22) In addition to the caret movement, this command will also extend the current selection to the new position.

PgUp (23) This command will move the caret one view up when it is located on the top line currently in the view. When the caret is not located on the top line of the view, it will be moved there.

Shift-PgUp	(24)	In addition to the caret movement this command will also extend the current selection to the new position.
PgDn	(25)	This command will move the caret one view down when it is located on the bottom line currently in the view. When the caret is not located on the bottom line of the view, it will be moved there.
Shift-PgDn	(26)	In addition to the caret movement, this command will also extend the current selection to the new position.

- **View commands**

Ctrl-Down	(80)	This command will scroll the view one line up. The command will always keep the caret inside the view. When it reaches the bottom of the view, the caret is automatically moved to the next line.
Ctrl-Up	(81)	This command will scroll the view one line down. The command will always keep the caret inside the view. When it reaches the top of the view, the caret is automatically moved to the previous line.
Ctrl-PgDn	(82)	This command will scroll the view one column to the left. The caret is not moved.
Ctrl-PgUp	(83)	This command will scroll the view one column to the right. The caret is not moved.

- **Edit commands**

Ctrl-Z	(100)	This command will undo the last change made to the edit control contents. You can undo any number of changes made to the control contents up to the maximum number of undo/redo hops.
Ctrl-Y	(101)	This command will redo the last change you have undone. You can re-do any number of changes up to the number of changes undone.
Shift-Del	(103)	This command will remove all characters to the right of the current caret position.
Backspace	(104)	This command will remove the character to the left of the caret. When the caret is located at the start of the line, the characters right of the caret are appended to the previous line and the caret is moved to be positioned between the old line contents and the appended characters.
Delete	(105)	This command removes the character to the right of the caret. When there are no characters to the right of the caret, the contents of the next line is appended to the current line.
Alt-Delete	(106)	This command deletes the current line.
Return	(107)	This command will split the current line and create a new line of the characters, if any, right of the caret. The caret is moved to the start of the newly created line.
Alt-U	(108)	This command will convert all lower-case characters of the word under the caret to upper-case characters. If the caret is not located on a word, nothing happens.
Alt-L	(109)	This command will convert all upper-case characters of the word under the caret to lower-case characters. If the caret is not located on a word, nothing happens.
Alt-S	(110)	This command will convert all lower-case characters to upper-case characters and upper-case characters to lower-case characters of the word under the caret. If the caret is not located on a word, nothing happens.
Ctrl-Delete	(113)	When the caret is located on a word, this command will delete all characters in the word right of the caret position.
Ctrl-Backspace	(114)	When the caret is located on a word, this command will delete all characters in the word left of the caret position.
Tab	(115)	This command does one of the two following things:

- When there is a valid text selection, this command will indent the lines covered by the selection right by one tab-stop.
- When there is no text selection, a tab is inserted at the current caret position.

Shift-Tab (116) This command does one of the two following things:

- When there is a valid text selection, this command will indent the lines covered by the selection left by one tab-stop.
- When there is no text selection, the caret is moved back to the previous tab-stop position.

Shift-Alt-T (117) This command will swap the line on which the caret is located with the previous line. When the caret is located on the first line, nothing will happen.

Shift-Ctrl-U (118) When there is a valid selection, this command will convert all lower-case characters in the selection to upper-case characters. If there is no valid selection, nothing happens.

Ctrl-U (119) When there is a valid selection, this command will convert all upper-case characters in the selection to lower-case characters. If there is no valid selection, nothing happens.

Ctrl-Alt-U (120) When there is a valid selection, this command will convert all lower-case characters in the selection to upper-case characters and all upper-case characters in the selection to lower-case. If there is no valid selection, nothing happens.

Ctrl-D (121) This command will insert the system date at the current caret location. The date output is formatted using the user's default format.

Shift-Ctrl-D (122) This command will insert the system time at the current caret location. The time output is formatted using the user's default format.

Ins (123) This command will toggle the current editing mode between overwrite and insert.

- **Mark / Clipboard commands**

Shift-Ctrl-W (200) This command will select the word on which the caret is currently located. When the caret is not located on a word, nothing happens.

Ctrl-A (201) This command will select all the text.

Shift-Ctrl-L (203) This command will select the line on which the caret is currently located.

Ctrl-V (225) This command will, when there is text present in the clipboard, paste the clipboard contents at the current position.

Ctrl-C (226) This command will, when there is a selection, copy the selected text to the clipboard.

Ctrl-X (227) This command will, when there is a selection, copy the selected text to the clipboard and remove the selection from the text.

Shift-Ctrl-C (229) This command will copy the line on which the caret is located to the clipboard.

Shift-Ctrl-X (230) This command will copy the line on which the caret is located to the clipboard and remove the line from the text.

Alt-C (232) This command will, if there is a selection, append the selection to the current clipboard contents. The result is placed on the clipboard again.

Alt-X (233) This command will, if there is a selection, append the selection to the current clipboard contents. The result is placed on the clipboard again, and the selection is removed from the text.

- **Search commands**

Ctrl-F3 (275) This command will find the next occurrence of the word under the caret. When the next occurrence is found, it is selected.

F3	(277)	This command will find the next occurrence of the current search pattern. When the search pattern is found, it is selected.
Shift-F3	(278)	This command will find the previous occurrence of the current search pattern. When the search pattern is found, it is selected.
Ctrl-]	(279)	This command will find the matching closing bracket if the caret is located on an opening bracket, or the matching open bracket if the caret is located on a closing bracket.
Shift-Ctrl-]	(280)	This command will find the matching closing bracket if the caret is located on an opening bracket, or the matching opening bracket if the caret is located on a closing bracket. In addition the text from the opening bracket up to and including the closing bracket is selected.
Ctrl-G	(300)	This command will show the *goto* dialog.
Ctrl-F	(301)	This command will show the *find* dialog.
Ctrl-H	(302)	This command will show the *replace* dialog.

- **File commands**

Ctrl-S	(401)	This command will save the control contents using its current name
Shift-Ctrl-Del	(402)	This command will delete all text. If the text has been modified the user will be prompted to save the contents first.
Ctrl-P	(403)	This command will open the printer properties dialog.

- **Bookmark commands**

Ctrl-F2	(252)	This command will clear the bookmark on the current line if it is set, or set the bookmark if it is cleared.
Shift-Ctrl-F2	(253)	This command will clear all bookmarks.
F2	(254)	This command will place the caret on the next line which has a bookmark set. When there is no next line with a bookmark, the text is searched starting at the first line.
Shift-F2	(255)	This command will place the caret on the previous line which has a bookmark set. When there is no previous line with a bookmark, the text is searched from the last line up again.

- **Breakpoint commands**

Ctrl-B	(262)	This command will toggle a breakpoint on the current line.
Ctrl-Shift-F9	(263)	This command will clear all breakpoints.

Options:

/A Start with the alias window open.

/B Start with the batch variables window open.

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

/E Start with the environment variables window open.

/F Start with the user-defined functions window open.

/I Load the batch file, but do not start execution

/K Keep the batch debugger window open and switch to edit mode after the batch file exits.

/N By default, the debugger window hides itself while the debugged batch file is executing. **/N** keeps that window visible in the background (the main ***4NT*** or ***TC*** window is brought to the foreground).

/W Open the Watch window when the debugger starts.

7.10 BEEP

Purpose: Beep the speaker or play simple music.

Format: BEEP [frequency duration ...] [asterisk | exclamation | hand | question | ok]

frequency	The beep frequency in Hertz (cycles per second).
duration	The beep length in 1/18th second intervals.
asterisk	Plays the system default "asterisk" sound.
exclamation	Plays the system default "exclamation" sound.
hand	Plays the system default "hand" sound.
question	Plays the system default "question" sound.
ok	Plays the system default "ok" sound.

See also: BeepFreq 107, BeepLength 107.

Usage:

BEEP generates a sound through your computer's speaker. You can use it in batch files to signal that an operation has been completed, or that the computer needs attention.

Because BEEP allows you to specify the frequency and duration of the sound, you can also use it to play simple music or to create different kinds of signals for the user.

You can include as many frequency and duration pairs as you wish. No sound will be generated for frequencies less than 20 Hz, allowing you to use BEEP as a way to create short delays. The default value for *frequency* is 440 Hz; the default value for *duration* is 2.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

This batch file fragment runs a program called *DEMO*, then plays a few notes and waits for you to press a key:

```
demo
beep 440 4  600 2  1040 6
pause Finished with the demo - hit a key...
```

The following table gives the *frequency* values for a five octave range (middle C is 262 Hz):

C	131	262	523	1046	2096
C# / Db	139	277	554	1108	2217
D	147	294	587	1175	2349
D# / Eb	156	311	622	1244	2489
E	165	330	659	1318	2637
F	175	349	698	1397	2794
F# / Gb	185	370	740	1480	2960

G	196	392	784	1568	3136
G# / Ab	208	415	831	1664	3322
A	220	**440**	880	1760	3520
A# / Bb	233	466	932	1866	3729
B	248	494	988	1973	3951

7.11 BREAK

Purpose: Enable or disable Ctrl-C and Ctrl-Break

Format: BREAK [ON | OFF]

Usage:

BREAK OFF will disable all **Ctrl-C** and **Ctrl-Break** handling in the command processor (though not necessarily in child processes). In CMD.EXE, BREAK OFF doesn't actually do anything, so setting it in ***4NT*** or ***TC*** will introduce a possible incompatibility with existing batch files.

7.12 BREAKPOINT

Purpose: Set a batch debugger breakpoint on the current line

Format: BREAKPOINT

Usage:

If the batch debugger is active, BREAKPOINT sets a breakpoint on the current line, stopping a "Step Out" sequence. If the batch debugger is not active, BREAKPOINT is ignored.

7.13 CALL

Purpose: Execute one batch file from within another.

Format: CALL file | :label [p1 [p2 ...]]

file	The batch file to execute.
:label	A label 566 in the current batch file.
p1, p2,...	Parameters for the batch file or subroutine

See also: CANCEL 211 and QUIT 333.

Usage:

Calling other batch files

CALL allows batch files to call other batch files (batch file nesting). The calling batch file is suspended while the called (second) batch file runs. When the second batch file finishes (without executing the CANCEL command), execution of the original batch file resumes at the next command.

WARNING! If you execute a batch file from inside another batch file without using CALL, the original batch file is terminated before the other one starts**.** This method of invoking a batch file from another is usually referred to as chaining. Note that if the batch file *A.BTM* uses **CALL B**, and *B.BTM* chains to the batch file *C.BTM***,** on exit from *C.BTM* (without executing a CANCEL 211 command) processing of batch file *A.BTM* is resumed as if it had used **CALL C**.

File *A.BTM:*

```
...
call b
echo xxx
```

File *B.BTM:*

```
...
c
```

File *C.BTM:*

```
...
quit
```

In the example above, after execution of the QUIT 333 command in *C.BTM* the **ECHO xxx** command in *A.BTM* is executed next.

The following batch file fragment compares an input line to ***wp*** and calls another batch file if it matches:

```
input  Enter your choice:  %%option
if "%option" == "wp" call wp.bat
```

Batch files may be nested up to 32 levels deep.

The current ECHO state is inherited by a called batch file.

The called batch file should always either return (by executing its last line, or by using the QUIT 333 command), or it should terminate batch file processing with CANCEL 211. Do not restart or CALL the original batch file from within the called file as this may cause an infinite loop or a stack overflow.

Calling a label

To provide compatibility with CMD.EXE, which does not support the GOSUB 280 command for subroutines in the same batch file, you may create a subroutine starting with a label 566 and terminated by any of the following:

- the end of the batch file
- QUIT 333
- EXIT 262
- CANCEL 211

Note that the last two do NOT return control to the CALL command. Do not use the RETURN 340 command!

Parameters passed to the subroutine are accessible as **%1**, **%2**, etc., in the same manner as in a batch file.

Exit code

CALL returns an exit code which matches the batch file return code. You can test this exit code with the % ? 410 or %? 409 environment variables, and use it with conditional commands 65 (**&&** and **||**).

See also GOSUB 280 and user-defined functions 275.

7.14 CANCEL

Purpose: Terminate batch file processing.

Format: CANCEL [value]

value The numeric exit code to return to the command processor.

See also: CALL 209 and QUIT 333.

Usage:

The CANCEL command ends all batch file processing, regardless of the batch file nesting level. Use QUIT 333 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 286 command, and the %? 409 variable). Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

7.15 CD / CHDIR

Purpose: Display or change the current directory.

Format: CD [/D /N /X] [path | -]

path The directory to change to, optionally including a drive letter

/D 213 (rive) /X 213 (exclude)
/N 213 (o extended search)

See also: CDD 213, MD 305, PUSHD 331, RD 333, and Directory Navigation 12.

Internet: Can be used with FTP Servers 42.

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 .

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 and 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 ***path*** to force an extended directory search. Read the section on Directory Navigation for complete details on these and other directory navigation features.

CD saves the current directory before changing to a new directory. You can switch back to the previous directory by entering CD -. (There must be a space between the CD command and the hyphen.) You can switch back and forth between two directories by repeatedly entering CD -. The saved directory is the same for both the CD and CDD commands. Drive changes and automatic directory changes also modify the saved directory, so you can use CD - to return to a directory that you exited with an automatic directory change.

Directory changes made with CD are recorded in the directory history list and can be displayed in the directory history window, which allows you to return quickly to a recently-used directory.

You can also use CD to display or change the current directory on an FTP server opened with IFTP. For example:

```
cd ftp:
ftp://jpsoft.com/

cd ftp:/pub
```

Note: FTP directory changes use neither the CDPATH feature nor the Extended Directory Searches database.

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.

Options:

/D Changes the current drive as well as directory. This option is included only for compatibility with the same option available in some versions of CMD.EXE. In most cases you should use CDD 213, which performs the same function.

/N Skips the standard extended directory search 14 when the directory is not found. This option is useful in batch files to force an error (rather than an extended search) if a directory is not found.

/X Don't save the current directory to the Directory History list.

7.16 CDD

Purpose: Change the current disk drive and directory.

Format: CDD [/A /D[drive …] /N /S[drive …] /U[drive…] /X] [*path* | -]

path The name of the directory (or drive and directory) to change to.
drive A drive or list of drives to include in the extended directory search database.

/A(ll drives) 214
/D(elete from JPSTREE.IDX) 214
/N(o extended search) 214
/S(earch tree) 214
/U(pdate tree) 215
/X (exclude from JPSTREE.IDX) 215

See also: CD 211, MD 305, PUSHD 331, RD 333, and Directory Navigation 12.

Usage:

CDD is similar to the CD 211 command, except that it also changes the default disk drive if one is specified. For example, to change from the root directory on drive A to the subdirectory *C:\WP:*

```
[a:\] cdd c:\wp
[c:\wp]
```

If no drive / path argument is supplied, CDD displays the current drive and directory.

CDD can also be used to create and update the Extended Directory Search 14 database (*JPSTREE.IDX*).

You can change to the parent directory with **CDD ..**; you can also go up one additional directory level with each additional [**.**]. For example, **CDD ....** will go up three levels in the directory tree.

CDD can also change to a network drive and directory specified with a UNC name (see File Systems 521 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 13, then the extended directory search 14 database in order to find a matching directory and switch to it. You can

disable this default extended search with **/N**. You can also use wildcards in the ***path*** to force an extended directory search. Read the section on Directory Navigation 12 for complete details on these and other directory navigation features.

CDD saves the current drive and directory before changing to a new directory. You can switch back to the previous drive and directory by entering **CDD -.** (There must be a space between the CDD command and the hyphen.) You can switch back and forth between two drives and directories by repeatedly entering **CDD -**. The saved directory is the same for both the CD and CDD commands. Drive changes and automatic directory changes 17 also modify the saved directory, so you can use **CDD -** to return to a directory that you exited with a drive change or an automatic directory change.

Directory changes made with CDD are recorded in the directory history list and can be displayed in the directory history window 62, which allows you to return quickly to a recently-used directory.

Windows limits the permissible length of the full subdirectory name (see the Directories and Subdirectories 522 topic for information on directory names).

When changing directories, ***4NT*** and ***TC*** normally maintain the original case of each path element. This is necessary for a few programs which are case-sensitive in their use of directory names. If you do not use such a program, disabling this case preservation will speed up directory changes slightly; to do so, see the SaveDirCase 141 directive.

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 14 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 14 when the directory is not found. This option is useful in batch files to force an error – rather than an extended search – if a directory is not found.

/S Builds or rebuilds the Extended Directory Search 14 database (*JPSTREE.IDX)*. You cannot move to a new drive and directory and use **/S** in the same command.

To include all local hard drives in the database, use the command:

```
cdd /s
```

To limit or add to the list of drives included in the database, list the drives and network volume names after the **/S** switch. For example, to include drives C, D, E, and the network volume \\server\dir1 in the database, use this command:

```
cdd /s cde \\server\dir1
```

All non-hidden directories on the listed drives will be indexed. CDD /S will also index the hidden directories if the CompleteHidden 110 .INI directive is set. Each time you use **/S**, everything in the previous directory database is replaced by the new database that is created. To update the database see **/U** below.

You can index specific subdirectories rather than an entire drive. For example, to index all directories on drive C but only the MSSDK directory tree on drive D:

```
cdd /s c:\ d:\mssdk
```

/U Updates the Extended Directory Search 14 database (*JPSTREE.IDX)* with the specified drives and directories instead of rebuilding the whole directory database. Uses the same syntax for drive and directory names as **/S**. For example, to update the *D:\MSSDK* tree and all of drive E:

```
cdd /u d:\mssdk e:\
```

/X Don't save the current directory to the Directory History list.

Note: The TREEEXCLUDE 400 variable can be used to specify which drives and directories should be ignored when updating the directory database.

7.17 CHCP

Purpose: Display or change the current system code page

Format: CHCP [*n*]

n A system code page number.

Usage:

Code page switching allows you to select different character sets for language support.

If you enter CHCP without a number, the current code page is displayed.

```
chcp
Active code page: 437
```

If you enter CHCP plus a code page number, the code page is changed. For example, to set the code page to multilingual:

```
chcp 850
```

When you use CHCP under Windows it only affects the current process, and any new programs started from within that process; the active code page in other processes remains unchanged.

7.18 CLS

Purpose: Clear the window and move the cursor to the upper left corner; optionally change the default display colors.

Format: CLS [/C /S] [[BRIght] *fg* ON [BRIght] *bg*

fg The new foreground color
bg The new background color

/C(lear buffer) 216 /S(croll buffer) 216 ***(4NT)***

Usage:

CLS can be used to clear the window without changing colors, or to clear the window and change the colors simultaneously, or to clear the entire 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 552 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 ***(4NT)*** 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.

7.19 COLOR

Purpose: Change the default display colors.

Format: COLOR [BRIght] *fg* ON [BRIght] bg

fg The new foreground color
bg The new background color

See also: CLS 215 and Colors and Color Names 552 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
```

4NT also supports the CMD.EXE `syntax`:

```
COLOR bf
```

In this syntax, ***b*** is a hexadecimal digit that specifies the background color and ***f*** is a hexadecimal digit that specifies the foreground color.

7.20 COPY

Purpose: Copy data between disks, directories, files, or physical hardware devices (such as your printer or serial port).

Format: COPY [/I"text"] [/A:... /C /D /E /F /G /H /J /K /L /M /N[est] /O /P /Q /R /S[n] /T /U /V /X /Z] [

@file] source [+] ... [/A|/B] [TO:] target [...] [/A|/B]

source	A file or list of files or a device to copy from.
target	A file, directory, or device to copy to
@file	A text file containing the names of the source files, one per line (see @file lists 32 for details)

/A(SCII) copy 221
/A:... (Attribute select) 221
/B(inary copy) 221
/C(hanged source files) 222
/D (Copy encrypted files) 222
/E (No error messages) 222
/F (No empty subdirectories) 222
/G (Display percentage) 222
/H (Include hidden files) 222
/I"text" (Match description) 222
/J (Restartable) 222
/K (Keep read-only attribute) 222
/L (ASCII FTP transfer) 222
/M(odified files 222
/N (Disable) 222
/O(nly if no target) 223
/P(rompt) 223
/Q(quiet) 223
/R(eplace) 223
/S(ubdirectories) 223
/T(otals) 223
/U(pdate target) 223
/V(erify) 223
/X (Clear archive) 223
/Z (overwrite) 223

See also: ATTRIB 198, MOVE 308, and REN 337.

File Selection

Supports attribute switches 28, extended wildcards 19, ranges 22, multiple file names 29, delayed variable expansion 31, and include lists 30. 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 31 for details.

Internet

Can be used with FTP / FTPS / TFTP / HTTP / HTTPS Servers 42.

Usage

The simplest use of COPY is to make a copy of a file, like this example which makes a copy of a file called *FILE1.ABC*:

```
copy file1.abc file2.def
```

You can also copy a file to another drive and/or directory. The following command copies ***FILE1*** to the *\MYDIR* directory on drive **E**:

```
copy file1 e:\mydir
```

When you COPY files to or from an LFN drive, you must quote any file names which contain white space or special characters.

To emulate an approach used by some implementations of CMD.EXE, see the COPYCMD 401 topic.

- **Copying Files**

You can copy several files at once by using wildcards 19:

```
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 30, 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 36 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, including our own **ftp://jpsoft.com**, use case sensitive file systems. For example:

```
copy ftp://ftp.abc.com/xyz/index index
```

Files copied to or from FTP/HTTP Servers are normally transferred in binary mode. To perform an ASCII transfer use the /L 222 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 222 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 288 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 42 and IFTP 288.

- **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 526 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 22 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: 221 Forces COPY to process ***source*** files with the attributes you specify after the :, or to process all ***source*** files regardless of attributes, if /A: 221 is used by itself.

/H 222 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 222 Retains the read-only attribute from each ***source*** file when creating the ***destination*** file. See /K 222 below for a special note if you are running under Novell NetWare.

/Z 223 Forces COPY to overwrite an existing ***destination*** file regardless of its attributes.

Use caution with /A: 221, /H 222, or /K 222 when both the ***source*** and ***destination*** directories contain file descriptions. If the ***source*** file specification matches the description file name (normally **DESCRIPT.ION**), and you use a switch which tells COPY to process hidden files, the **DESCRIPT.ION** file itself will be copied, overwriting any existing file descriptions in the ***destination*** directory. For example, if the *\DATA* directory contains file descriptions this command would overwrite any existing descriptions in the *\SAVE* directory:

```
[c:\data] copy /h d*.* \save\
```

If you remove the hidden attribute from the **DESCRIPT.ION** file, the same caution applies even if you do not use /A: 221, /H 222, or /K 222, as **DESCRIPT.ION** is then treated like any other file.

You can copy files to multiple destinations with the TO: option. For example, to copy ***letter.doc*** to three different directories:

```
copy letter.doc TO: \save\ f:\backups\ q:\letters\
```

Note: The wildcard expansion process will attempt to allow both CMD.EXE-style "*extension*" matching (assumes only one extension, at the end of the word) and the advanced ***4NT*** and ***TC*** string matching (allowing things like ****.*.abc***) when an asterisk is encountered in the ***destination*** of a COPY command.

COPY supports connected web folders. If an HTML file (i.e., with an .htm or .html extension) is copied, COPY will look for a folder in the same directory with the same name and an extension of ".files". If it is found, the .files directory will be copied to the target directory. You can disable connected web folders by setting the registry key:

HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\NoFileFolderConnection=0

Options

The /A 221 (ASCII copy) and /B 221 (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 221 or /B 221, 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 223 and /P 223, COPY will generally take a "conservative" approach and give priority to the option which generates more prompts or more information.

/A If you use **/A** with a ***source*** filename, the file will be copied up to, but not including, the first Control-Z (ASCII: 26) character in the file. If you use **/A** with a ***destination*** filename, a Control-Z will be added to the end of the file. **/A** is the default when appending files, or when the ***destination*** is a device like **NUL**, rather than a disk file.

This option applies to the filename immediately preceding it, and to all subsequent filenames until the file name preceding the next /A or /B 221 option.

/A:... Select only those files that have the specified attribute(s) set. See Attribute Switches 28 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 221 switch, above. Do not use **/A:** with ***@file*** lists. See @file lists 32 for details. Hidden or system files selected by this option overwrite hidden or system files.

/B If you use **/B** with a ***source*** filename, the entire file is copied; <SUB> characters, if any, in the file are considered ordinary data to be copied. Using **/B** with a ***destination*** filename prevents addition of a <SUB> to the end of the ***destination*** file. **/B** is the default unless source files are appended to the target file, or the target is a device, e.g., **NUL**.

This option applies to the filename immediately preceding it, and to all subsequent filenames until the file name preceding the next /A 221 or /B option.

/C Copy files only if the ***destination*** file exists and is older than the ***source*** (see also /U 223). 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 223. Do not use **/C** with ***@file*** lists. See @file lists 32 for details.

/D ***(***Windows XP+ Only***)*** Force copy of an encrypted file even when the target will be decrypted (for CMD.EXE compatibility).

/E (No error messages) Suppress all non-fatal error messages, such as **File not found** or **Can't copy file to itself**. Fatal error messages, such as `Drive not ready`, will still be displayed. This option is most useful in batch files and aliases.

/F When used with **/S**, COPY will not create any empty subdirectories.

/G Displays the percentage copied and the transfer rate (in Kbytes/second). Useful when copying large files across a network or via FTP to ensure the copy is proceeding. When /V 223 is also used, reports percentage verified.

/H Copy all matching files including those with the hidden and/or system attribute set. See the cautionary note under **Advanced Features** above before using /H when both ***source*** and ***destination*** directories contain file descriptions.

/I"*text*" (Match descriptions) Select ***source*** files by matching text in their descriptions. See Description Ranges 28 for details.

/J Copy the file in restartable mode. The copy progress is tracked in the destination file in case the copy fails. The copy can be restarted by specifying the same source and destination file names.

/K (Keep read-only attribute) To maintain compatibility with CMD.EXE, COPY normally maintains the hidden and system attributes, sets the archive attribute, and removes the read-only attribute on the target file. **/K** tells COPY to also maintain the read-only attribute on the ***destination*** file. However, if the ***destination*** is on a Novell NetWare volume, this option will fail to maintain the read-only attribute. This is due to the way NetWare handles file attributes, and is not a problem in COPY.

/L 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 223 switch, or use ATTRIB 198. Do not use /M with ***@file*** lists. See @file lists 32 for details.

/N Do everything except actually perform the copy. This option is useful for testing what the result of a complex COPY command will be. /N displays how many files would be copied. /N does not prevent creation of ***destination*** subdirectories when it is used with /S 223.

A **/N** with one of the following arguments has an alternate meaning:

- **e** Don't display errors.
- **s** Don't display the summary.
- **t** Don't update the CD / CDD extended directory search 14 database *(JPSTREE.IDX)*.

/O Only copy the source file if the target file doesn't exist.

/P Ask the user to confirm each *source* file. Your options at the prompt are explained in detail under Page and File Prompts 41. Note: the CopyPrompt 111 directive can be used to force prompting at the command line only. See also: the /Q 223 option below.

/Q Don't display filenames, percentage copied, total number of files copied, etc... When used in combination with the /P 223 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 223.

/R Prompt the user before overwriting an existing file. Your options at the prompt are explained in detail under Page and File Prompts 41. See also: the CopyPrompt 111 directive. (For compatibility with CMD.EXE, a /Y option on the command line is changed to /R.)

/S Copy the subdirectory tree starting with the files in the ***source*** directory plus each subdirectory below that. The ***destination*** must be a directory; if it doesn't exist, COPY will attempt to create it. COPY will also attempt to create needed subdirectories on the tree below the ***destination***, including empty ***source*** directories. If COPY /S creates one or more destination directories, they will be added automatically to the extended directory search 14 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 32 for details.

If you specify a number after the **/S**, COPY will limit the subdirectory recursion to that number. For example, if you have a directory tree "\a\b\c\d\e", **/S2** will only affect the "a", "b", and "c" directories.

/T Turns off the display of filenames, like /Q 223, but does display the total number of files copied.

/U Copy each ***source*** file only if it is newer than a matching ***destination*** file or if a matching ***destination*** file does not exist (see also /C 222). 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 32 for details. When used with file systems that have different time resolutions (such as FAT 563 and NTFS 567), **/U** will attempt to use the "coarsest" resolution of the two.

/V Verify each disk write by performing a true byte-by-byte comparison between the source and the newly-created target file. This option will significantly increase the time necessary to complete a COPY command.

/X Clear the archive attribute from the source file after a successful copy. This option is most useful if you are using COPY to maintain a set of backup files.

/Z Overwrite ***destination*** files regardless of their attributes. Without this option, COPY will fail with an "Access denied error" if the ***destination*** file has its read-only attribute set, or (depending on other options) its hidden or system attribute set. Required to overwrite read-only targets regardless of other options. Required to overwrite hidden or system targets unless the source also has the attribute, and either /H 222 or /A: 221 is used to select it.

7.21 DATE

Purpose: Display and optionally change the system date.

Format: DATE [/T] [*mm -dd -yy*]

mm	The month (1 - 12)
dd	The day (1 - 31)
yy	The year (80 - 99 or a 4- digit year)

/T (Display only) 224

See also: TIME 378.

Usage:

If you simply type DATE without any parameters, you will see the current system date and time, and be prompted for a new date. Press **Enter** if you don't wish to change the date. If you type a new date, it will become the current system date, which is included in the directory entry for each file as it is created or altered:

```
date
Fri  Sep 1, 2006  9:30:06
Enter new date (mm-dd-yy):
```

You can also enter a new system date by typing the DATE command plus the new date on the command line:

```
date 11-16-2006
```

You can use hyphens, slashes, or periods to separate the month, day, and year entries. The year can be entered as a 2-digit or 4-digit value. Two-digit years between 80 and 99 are interpreted as 1980 - 1999; values between 00 and 79 are interpreted as 2000 - 2079.

DATE adjusts the format it expects depending on your country settings. When entering the date, use the correct format for the country setting currently in effect on your system.

You can also use the international date format **yyyy-mm-dd**.

Option:

/T	Displays the current date but does not prompt you for a new date. If a new date is specified in the same command as **/T** the new date will be ignored.

7.22 DDEEXEC

Purpose: Send a DDE command to another application.

Format: DDEEXEC server, topic, command

server	The DDE name of the program that will receive the command.
topic	The server's topic name for receiving the command.
command	The command string to send to the server.

Usage:

Windows supports a form of communication between programs called Dynamic Data Exchange or DDE. Using DDE, one program can send a command or data to another. The receiving program is usually called the DDE server; the sending program is usually called the DDE client. When you use DDEEXEC, ***TC*** acts as a DDE client and the program which receives the command is the server. (***TC*** can also act as a **DDE server**. See the DDE Support 87 topic for details.)

For example, if you wanted to instruct Internet Explorer to open JP Software's web site, you could use this command:

```
ddeexec iexplore, WWW_OpenURL, "http://jpsoft.com/"
```

In this example, the ***server*** name is **iexplore**, the ***topic*** name is **WWW_OpenURL**, and the ***command*** is **"http://jpsoft.com/"**. (Internet Explorer must be running, and the version on your system must support this syntax, for this example to work.)

The server name, topic name, and possible commands are defined by the server application. See the documentation included with the programs to which you want to send DDE messages for details about the names and commands to use.

7.23 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.

7.24 DEL / ERASE

Purpose: Erase one file, a group of files, or entire subdirectories.

Format: DEL [ranges] [/A:[[-|+]rhsadecijopt /E /F /I"text" /K /N[est] /P /Q /R /S[n] /T /W /X /Y /Z] [@file] file...

file The file, subdirectory, or list of files or subdirectories to erase.
@file A text file containing the names of the files to delete, one per line (see @file lists 32 for details).

/A: 228 (Attribute select)
/E 228 (No error messages)
/F 228 (orce delete)
/I (match descriptions)
/K 228 (no Recycle Bin)
/N 228 (Disable)
/P 228 (rompt)
/Q 228 (uiet)
/R 228 (ecycle bin)
/S 228 (ubdirectories)
/T 228 (otal)
/W 228 (ipe)
/X 229 (remove empty subdirectories)
/Y 229 (es to all prompts)
/Z 229 (ap hidden and read-only files)

File Selection

Supports attribute switches 28, extended wildcards 19, ranges 22, multiple file names 29, and include lists 30. Use wildcards with caution on LFN volumes; see LFN File Searches 31 for details.

Internet

Can be used with FTP/HTTP Servers 42.

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 566, 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 27. For example, to delete all files in the current directory except those whose extension is *.TXT*, use a command like this:

```
del /[!*.TXT] *.*
```

When using exclusion ranges or other more complex options you may want to use the **/N** switch first, to preview the effects of the DEL without actually deleting any files.

If you enter a subdirectory name, or a filename composed only of wildcards (* and/or ?), DEL asks for confirmation (Y or N) unless you specified the /Y option. If you respond with a Y, DEL will delete all the files in that subdirectory (hidden, system, and read-only files are only deleted if you use the /Z option). NOTE: The Windows command processor, CMD.EXE, behaves the same way but does not ask for confirmation if you use /Q to delete files quietly. If you want the command processor to follow CMD.EXE's approach and skip the confirmation prompt when /Q is used, set DelGlobalQuery 113 to No in the .INI file 91. Use caution if you set DelGlobalQuery 113 to No, as this will allow DEL /Q to delete an entire directory without prompting for confirmation.

DEL displays the amount of disk space recovered, unless the /Q option is used (see below). It does so by comparing the amount of free disk space before and after the DEL command is executed. This amount may be incorrect if you are using a deletion tracking system which stores deleted files in a hidden directory, or if another program performs a file operation while the DEL command is executing.

Remember that DEL removes file descriptions along with files. Most deletion tracking systems will not be able to save or recover a file's description, even if they can save or recover the data in a file. This applies to the use of DEL with the Windows Recycle Bin, too - the description will be lost.

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 228 option, which overwrites the file with zeros before deleting it. Use this option with caution. Once a file is obliterated, it is impossible to recover. Remember: /W 228 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 % ? 409 internal variable, and use it with conditional commands 65 (**&&** and **||**).

Use caution when using wildcards with DEL on LFN drives, because the command processor's wildcard matching can match both short and long filenames. This can delete files you did not expect; see LFN File Searches 31 for additional details.

- **Recycle Bin**

When you delete files with DEL, ***4NT*** and ***TC*** do not move the deleted files to the Windows Recycle Bin by default. You can change this default on the "Startup" tab of the configuration dialog, or with the RecycleBin 139 directive in the .INI file 91. If you have disabled the recycle bin, you can override the setting and place deleted files in the recycle bin with the /R 228 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 228 option:

```
del /k letters.txt
```

You can also exclude files from the Recycle bin, even if **RecycleBin=Yes** is set, or if the command use the /R 228 option, with the RecycleExclude 400 environment variable.

- **FTP Usage**

If you have appropriate permissions, you can delete files on FTP servers 42. For example:

```
del ftp://ftp.abc.com/index
```

You can also use the IFTP 288 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 526 for additional details.

Options

/A: Delete only those files that have the specified attribute(s) set. See Attribute Switches 28 for information on the attributes which can follow **/A:**. Do not use /A: with @file lists. See @file lists 32 for details.

/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 229 (see below): it deletes read-only, hidden, and system files as well as normal files.. It is included for compatibility with CMD.EXE.

/I"text" Select filenames by matching text in their descriptions. The text can include wildcards 19 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. This option overrides the default RecycleBin=Yes 139 *.INI* setting.

/N Do everything except actually delete the file(s). This is useful for testing the result of a DEL.

A **/N** with one of the following arguments has an alternate meaning:

- **e** Don't display errors
- **s** Don't display the summary
- **t** Don't update the CD / CDD extended directory search 14 database *(JPSTREE.IDX)*

/P Prompt the user to confirm each erasure. Your options at the prompt are explained in detail under Page and File Prompts 41.

/Q Don't display filenames as they are deleted, or the number of files deleted or bytes freed. If DelGlobalQuery 113 is set to **No** in the INI file 91 then **/Q** also disables the normal confirmation prompt when performing wildcard deletions (*e.g.* DEL *.*), for compatibility with CMD.EXE. Use caution if you set DelGlobalQuery 113 to No, as this will allow DEL /Q to delete an entire directory without prompting for confirmation. See also /T 228.

/R Delete files to the Windows Recycle Bin. This option overrides a RecycleBin=No 139 *.INI* setting.

/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 32 for details.

If you specify a number after the /S, DEL will limit the subdirectory recursion to that number. For example, if you have a directory tree "\a\b\c\d\e", /S2 will only affect the "a", "b", and "c" directories.

/T Don't display filenames as they are deleted, but display the total number of files deleted plus the amount of free disk space recovered. Unlike /Q 228, the /T option will not speed up deletions under DOS.

/W Clear the file to zeros before deleting it. Use this option to completely obliterate a file's

contents from your disk. Once you have used this option it is impossible to recover the file even if you are using an undelete utility, because the contents of the file are destroyed before it is deleted. /W overwrites the file only once; it does not adhere to security standards which require multiple overwrites with varying data when destroying sensitive information. /W will override a /R 228 or a RecycleBin = Yes 139 *.INI* setting.

/X Removes empty subdirectories (only useful when used with /S 228). If DEL deletes one or more directories, they will be removed automatically from the extended directory search database 14.

/Y The reverse of /P 228 — it assumes a Y response to everything, including deleting an entire subdirectory tree. The command processor normally prompts before deleting files when the name consists only of wildcards or a subdirectory name (see above); /Y overrides this protection and should be used with extreme caution!

/Z 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 260 works by hiding files, /Z will override an EXCEPT 260 command. However, files specified in a file exclusion range 27 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\
```

7.25 DELAY

Purpose: Pause for a specified length of time.

Format: DELAY [/B /M *time*]

time The number of seconds or milliseconds to delay.

/B(reak enabled) 230 /M(illiseconds) 230

Usage:

DELAY is useful in batch file loops while waiting for something to occur. For example, to wait for 10 seconds:

```
delay 10
```

DELAY is most useful when you need to wait a specific amount of time for an external event, or check a system condition periodically. For example, this batch file checks the battery status (as reported by your Advanced Power Management drivers) every 15 seconds, and gives a warning when battery life falls below 30%:

```
do forever
   iff %_apmlife lt 30 then
      beep 440 4 880 4 440 4 880 4
      echo Low Battery!!
   endiff
   delay 15
enddo
```

The ***time*** value can be as large as about 1 billion seconds (34 years!). If you don't enter a ***time***, the default is 1 second.

The command processor uses the minimum possible processor time during a DELAY, in order to allow other applications full use of system resources.

You can cancel a delay by pressing **Ctrl-C** or **Ctrl-Break**.

Options:

/B Allows terminating a DELAY by pressing a key.

/M Count by milliseconds instead of seconds. Normally only used for delays of less than 1 second.

7.26 DESCRIBE

Purpose: Create, modify, or delete file and subdirectory descriptions.

Format: Creating or modifying descriptions
DESCRIBE [ranges... /I"text"] [/A:atrlst] [@file] file [[/D]"description"]] ...]

Description file updating
DESCRIBE /U [[d:\path\descript.ion] ...]]

file	The file or files to operate on.
@file	A text file containing the names of the files to describe, one per line (see @file lists 32 for details).
"description"	The description to attach to the file.

/A: 232 (Attribute select) /I 232 (match description)
/D 232 (escription follows) /U 232 (pdate) descriptions file

See also: @DESCRIPT 445, DIR 233, and SELECT 345

File Selection

Supports attribute switches 28, extended wildcards 19, ranges 22, multiple file names 29, and include lists 30. Use wildcards with caution on LFN volumes; see LFN File Searches 31 for details.

Usage:

DESCRIBE adds descriptions to files and subdirectories. (Volume root directories cannot have descriptions.) The descriptions are displayed by DIR 233 in single-column mode and by SELECT 345, and can be retrieved using the @DESCRIPT 445 function. Descriptions let you identify your files in much more meaningful ways than you can in a filename alone.

(TC) You can also enter or modify descriptions with the Descriptions dialog 81. The dialog allows you to select a single file and modify its description using a dialog box, rather than using the DESCRIBE command. The information in this section also applies to descriptions created via the dialog, unless otherwise noted.

You enter a description on the command line by typing the DESCRIBE, 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 566 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, the command processor will treat the string as a quoted file name, not as description text as you intended. To resolve this problem use the **/D** 232 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 566 drives you will not see file descriptions in a normal DIR 233 display, because DIR 233 must leave space for the long filenames. To view the descriptions, use DIR 233 /Z to display the directory in FAT format. See DIR 233 for more details.

Each description can be up to 511 characters long. You can change this limit on **Misc** tab of the configuration dialogs, or with the DescriptionMax 115 directive in the .INI file 91. 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 233 output). DESCRIBE can edit descriptions longer than DescriptionMax 115 (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 NTFSDescriptions 134 directive in your .INI file), or in each directory in a hidden file called *DESCRIPT.ION*. Use the ATTRIB 198 command to remove the hidden attribute from this file if you need to copy or delete it. *DESCRIPT.ION* is always created as a hidden file, but will not be rehidden by the command processor if you remove the hidden attribute.

You can change the description file name with the DescriptionName 115 directive in the .INI file 91 or the SETDOS /D 354 command, and retrieve it with the %_DNAME 415 internal variable. Use caution when changing the description file name, as changing the name from the default will make it difficult to transfer file descriptions to another system.

The description file is modified appropriately whenever you perform an internal command which affects it (such as COPY 216, MOVE 308, DEL 225, or RENAME 337), but not if you use an external program (such as XCOPY or Explorer). You can disable description processing on the **Misc** tab of the configuration dialogs 151, with the Descriptions 115 directive in the .INI file 91, or with SETDOS /D 354.

When you COPY 216, MOVE 308 or REN 337 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 216 and MOVE 308 for additional details.

If you disable descriptions with the SETDOS 354 /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 28 for information on the attributes which can follow **/A:**. Do not use **/A:** with @file lists. See @file lists 32 for details.

/D"description**"**
The quoted string following the **/D** switch without any separation is used as a description, not a file name, avoiding ambiguity in the meaning of quoted strings. See the **Usage** section above for details.

/I"text**"**
Select files by matching text in their descriptions. The text can include wildcards 19 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 32 for details.

/U Update the *DESCRIPT.ION* file (or the file specified by the DescriptionName 115 directive), deleting the entries for any nonexistent files. If no filename is supplied, DESCRIBE will process *DESCRIPT.ION* in the current directory. Otherwise, DESCRIBE will process *DESCRIPT.ION* in the specified path(s). This option may not be used in conjunction with other DESCRIBE options.

7.27 DETACH

Purpose: Start a console (character-mode) application program in detached mode.

Format: DETACH [/Q] command

/Q(uiet) 233

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 364 and TASKEND 373.

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 36 symbols. In most cases, you should only DETACH text-mode programs, since most graphical applications cannot run without a screen or keyboard, or have their input and output redirected.

The ***command*** can be an internal command, external command, alias, or batch file. If it is not an

external command, the command processor will detach a copy of itself to execute the command.

For example, the following command will detach a copy of the command processor to run the batch file *XYZ.BTM*:

```
detach xyz.btm
```

You can also include any parameters or command line switches which the command knows how to interpret:

```
detach "xyz.btm Monday Nebraska"
```

Once the program has started, the command processor returns to the prompt immediately. It does not wait for a detached program to finish.

The Process ID of the detached program is returned in the _DETACHPID 415 variable.

You can use the TASKEND 373 command to stop a detached program which does not terminate on its own.

Options:

/Q Don't display the new process's ID.

7.28 DIR

Purpose: Display information about files and subdirectories.

Format: DIR `[ranges] [options] [file...]`

ranges one or more ranges 22
options one or more file selection or report format options 240
file The file, directory, or list of files or directories to display.

/1 240	1 column output	/L 241	Lower case
/2 240	2 column output	/M 241	suppress footer
/4 240	4 column output	/N 241 [eshv]	New format or disable options
/: 240	show streams	/O 242	Order
/A 241	Attribute select	/P 242	Pause
/B 241	Bare (name only)	/Q 242	show owner
/C 241	show Compression	/R 242	disable wrap
/D 241	Disable color coding	/S 242n	show Subdirectories to depth ***n***
/E 241	upper case	/T 242	show aTtribute
/F 241	Full path	/T: 243	time type
/G 241	allocated size	/U 243	show summary information
/H 241	Hide dots	/V 243	Vertical sort
/I 241 "text"	description range	/W 243	Wide
/J 241	Justify names	/X 243	show short names
/K 241	suppress header	/Z 243	use FAT format

See also: ATTRIB 198, DESCRIBE 230, PDIR 320, SELECT 345, and SETDOS 354.

File Selection

Supports extended wildcards 19, ranges 22, multiple file names 29, and include lists 30.

Internet: Can be used with FTP servers 42.

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 320 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 187 command.

For example, to display all the files in the current directory, in 2 columns, sorted vertically (down one column then down the next), and with a pause at the end of each page:

```
dir /2/p/v
```

To set up this format as the default, using an alias:

```
alias dir=*dir /2/p/v
```

When you use DIR on an LFN drive, you must quote any file names which contain white space or special characters.

The following sections group DIR's features together in several categories. Many of the sections move from a general discussion to more technical material. If you find some of the information in a category too detailed for your needs, feel free to skip to the beginning of the next section. The sections are:

Selecting Files

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 19. 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 30 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 230 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 22 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 D is APPS    Serial ...
 Directory of  D:\4NT\4NT\*

8-24-2005  12:17         <DIR>    .
8-24-2005  12:17         <DIR>    ..
8-21-2005  18:08         212,854  4NT.EXE
8-02-2005  10:08              45  4NT.INI
```

(See Switching Formats below for information on changing the standard long filename format to allow room for file descriptions.)

On FAT volumes which do not support long file names, the default DIR format contains 5 columns: the file name, the file size in bytes, the date of the last write, the time of the last write, and the file's description. File names are listed in lower-case; directory names in upper case:

```
Volume in drive C is C - BOOTUP    Serial ...
 Directory of  C:\*.*

.            <DIR>       8-24-05  12:17
..           <DIR>       8-24-05  12:17
TEST         <DIR>       8-01-05  16:21
jpstree.idx     96967    8-28-05  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 the command processor on an LFN drive:

	default or /1 240	**/2** or **/4 columns**	**/W (wide)**
Normal	date, time, size, LFN	2 - 4 columns, LFNs only	No. of columns based on longest LFN
/Z 243 (FAT)	truncated LFN, size, date, time	2 - 4 columns, truncated LFN plus date, time, size	5+ columns, truncated LFNs only
/X 243 (SFN)	date, time, size, SFN, LFN	2 - 4 columns, SFNs plus date, time, size	5+ columns, SFNs only
/X 243 /Z 243	SFN, size, date, time	(Same as /X)	(Same as /X)

Color-Coded Directories

DIR can display each file name and the associated file information in a different color, depending on the file's extension.

To choose the display colors, you must either use the SET 351 command to create an environment variable called COLORDIR 398, use the configuration dialogs 151, or use the ColorDir directive 109 in the .INI file 91. If you use neither the COLORDIR variable nor the ColorDir directive, DIR will use the default screen colors for all files.

If you use both the COLORDIR variable and the ColorDir directive, the environment variable will override the settings in your .INI file 91. You may find it useful to use the COLORDIR variable for experimenting, then to set permanent directory colors with the ColorDir directive.

The format for both the COLORDIR environment variable and the ColorDir directive is:

```
ext ... :ColorName; ...
```

where "ext" is either a file extension (which may include wildcards) or one 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	Files which are junctions
NORMAL	File with no attribute set
NOTINDEXED	Files whose content is not indexed
OFFLINE	Offline files
RDONLY	Read-only files
SPARSE	Sparse files
SYSTEM	System files
TEMPORARY	temporary files

and "ColorName" is any valid color name (see Colors and Color Names 552 for information on color names).

Note that if a file uses one of the reserved file type names shown above as its extension (e.g. *xyz.hidden*) , that file will receive the color defined for the file type.

Unlike most color specifications, the background portion of the color name may be omitted for directory colors. If you don't specify a background color, DIR will use the current screen background color.

For example, to display *.COM* and *.EXE* files in red on the current background, *.C* and *.ASM* files in bright cyan on the current background, read-only files in green on white, and everything else in the default color:

```
set colordir=com exe:red; c asm:bright cyan; rdonly:green on white
```

Extended wildcards 19 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.** 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 36 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 42. For example:

```
dir ftp://jpsoft.com/4nt
```

You can also use the IFTP 288 command to start an FTP session on a server, and then use a simplified syntax to specify the files and directories you want.

Other Notes

If you have selected a specific country code for your system, DIR will display the date in the format for that country. The default date format is U.S. (mm-dd-yy). The separator character in the file time will also be affected by the country code. Thousands and decimal separators in numeric displays are affected by the country code, and by the ThousandsChar and DecimalChar settings selected with the configuration dialogs or in the .INI file 91.

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.

Options:

Options on the command line apply only to the filenames which follow the option, and options at the end of the line apply to the preceding filename only. This allows you to specify different options for different groups of files, yet retains compatibility with the traditional DIR command when a single filename is specified.

/1 Single column display – display the filename, size, date, and time; also displays the description on drives which do not support long filenames. This is the default. If **/T** is used the attributes are displayed instead of the description; if **/C** or **/O:c** is used the compression ratio is displayed instead of the description. This option is most useful if you wish to override a default **/2**, **/4**, or **/W** setting stored in an alias.

/2 Two column display – display just the name (on LFN drives), or display the filename, size, date, and time on other drives. See **Multiple Column Displays** above for more details.

/4 Four column display – display just the name (on LFN drives); or display the filename and size, in K (kilobytes) or M (megabytes) on other drives, with files between 1 and 9.9 megabytes in size displayed in tenths (*i.e.*, "2.4M"). See **Multiple Column Displays** above for more details.

/: Display file stream names and sizes on NTFS volumes. When combined with the /B or /F options, the size is omitted.

When **/:** is used in conjunction with **/B** (Bare), the file name is displayed on the first line, then any streams, indented two spaces, on subsequent lines:

```
c:\test\myfile.dat
```

```
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 28 for information on the attributes which can follow **/A:**.

/B Suppress the header and summary lines, and display file or subdirectory names only, in a single column. This option is most useful when you want to redirect a list of names to a file or another program. If you use **/B** with **/S**, DIR will show the full path of each file (the same display as **/F**) instead of simply its name and extension. If you use **/B** with **/X** on an LFN drive, DIR will display the short name of each file instead of the long name. **/B** also sets **/H.**

/C Display per-file and total compression percentage on NTFS drives with compression enabled. **/C** only works in single-column mode; it is ignored if **/2**, **/4**, or **/W** is used.

/D Temporarily disable directory color coding. May be required when color-coded directories are used and DIR output is redirected to a character device like a serial port (*e.g.*, COM1). **/D** is not required when DIR output is redirected to a file.

/E Display filenames in upper case.

/F Display each filename with its drive letter and path in a single column, without other information. If you use **/F** with **/X**, the "short" version of the entire path is displayed.

/G Display the allocated disk space instead of the actual size of each file.

/H Suppress the display of the "**.**" and "**..**" directories.

/I"*text***"** Select filenames by matching text in their descriptions. The text can include wildcards 19 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:

e Don't display an error message if no files match.
h Don't display the header
s Don't display the summary.
v Don't display the volume information.

/O Set the sorting order. You may use any combination of the sorting options below. If multiple options are used, the listing will be sorted with the first sort option as the primary key, the next as the secondary key, and so on:

n Sort by filename and extension, unless **e** is explicitly included. This is the default.
- Reverse the sort order for the next sort key
a Sort names and extensions in standard ASCII order, rather than sorting by magnitude when numeric substrings are included in the name or extension.
c Sort by compression ratio (the least compressed file in the list will be displayed first). For single-column directory displays in the short filename format, the compression ratios will be used as the basis of the sort and will also be displayed. For wider displays (**/2**, **/4**, and **/W**) and displays in LFN format, the compression ratios will be used to determine the order but will not be displayed. For information on supported compression systems see **/C** above.
d Sort by date and time (oldest first); also see **/T:acw**
e Sort by extension
g Group subdirectories first, then files
i Sort by file description (ignored if **/C** or **/O:c** is also used)
o Sort by owner
r Reverse the sort order for all options
s Sort by size
u Unsorted

/P Wait for a key to be pressed after each screen page before continuing the display. Your options at the prompt are explained in detail under Page and File Prompts 41.

/Q Display the file or directory owner (NTFS and remote directories only).

/R Forces long descriptions to be displayed on a single line, rather than wrapped onto two or more lines. Use **/R** when output is redirected to a character device, such as a serial port or the printer; or when you want descriptions truncated, rather than wrapped, in the on-screen display.

/S Display file information from the current directory and all of its accessible subdirectories. DIR will only display headers and summaries for those directories which contain files that match the filename(s), ranges, and attributes that you specify on the command line. DIR will display hidden subdirectories for compatibility with **CMD.EXE**.

If you specify a number after the /S, DIR will limit the subdirectory recursion to that number. For example, if you have a directory tree "\a\b\c\d\e", /S2 will only affect the "a", "b", and "c" directories.

/T Display the filenames and attributes in the format **RHSADENTPCOI**, regardless of volume type:

R Read-only **A** Archive

H	Hidden	**D**	Subdirectory
S	System		
E	Encrypted	**C**	Compressed
N	Normal	**O**	Offline
T	Temporary	**I**	Not content-indexed
P	Sparse file	**J**	Junction (reparse point)

Attributes which are set are represented by their letter, unset attributes by the _ (underscore) character.

If you wish to add another option after **/T**, you must start the next option with a forward slash. If you don't, the command processor will interpret the **/T** as the **/T:{acw** time display switch (see below) and the following character as a time selector. For example:

```
dir /tz          incorrect, will display error
dir /t/z         correct
```

/T:a|c|w 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).

/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 /2 240, ***/4*** 240 or /W 243 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 the command processor window, using 16 characters in each column. Otherwise (*i.e.*, when long filenames are displayed) the number of columns depends on the width of the longest name in the listing. See **Multiple Column Displays** above for more details.

/X Display both the short name (8-character name plus 3-character extension) and the long name of each file on an LFN drive. In normal single-column output the short name is displayed first, followed by the long name. The short name column is left blank if the short name and long name are the same. 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 /2 240, ***/4*** 240, /B 241, /W 243 , 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 243 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 115 exists, /Z defaults to using the name of the . and .. directories as description for those entries

7.29 DIRHISTORY

Purpose: Display, add to, clear, or read the directory history list.

Format: DIRHISTORY [/A directory /F /G /L /N /P /R filename]

directory The name of a directory to be added to the directory history.
filename The name of a file containing entries to be added to the directory history.

/A 245(dd)
/F 245(ree)
/G 245(lobal)
/L 245(ocal)
/N 245(o duplicates)
/P 245(ause)
/R 245(ead)

See also: HISTORY 285.

Usage

Every time you change to a new directory or drive, the command processor save the previous directory in an internal directory history list. The directory history window 62 allows you to use the list to return to a previous directory. See also: directory navigation 12.

The DIRHISTORY command lets you view and manipulate the directory history list directly. If no parameters are entered, DIRHISTORY will display the current directory history list:

```
dirhistory
```

With the options explained below, you can clear the list, add new directories to the list without changing to them, save the list in a file, or read a new list from a file.

The number of directories saved in the directory history list depends on the length of each directory name. The list size can be specified at startup (see the History Options Tab 155 or the DirHistory 115 directive in the INI file 91). The default size is 1,024 characters.

Your directory history list can be stored either locally (a separate history list for each copy of the command processor) or globally (all copies of the command processor share the same list). For details see the discussion of local and global history lists 52. If you use global lists, SHRALIAS 361 can save the list when no copy of the command processor is active, as long as you do not restart Windows.

You can save the directory history list by redirecting the output of DIRHISTORY to a file. This example saves the history to a file called *DIRHIST* and reads it back again.

```
dirhistory > dirhist
    .....
dirhistory /r dirhist
```

Because the directory history stores each name only once, you don't have to delete its contents before reading back the file unless you want to delete the directories that were visited by the intervening commands.

If you need to save your directory history at the end of each day's work, you might use the first of these commands in your TCSTART 8 ***(TC)*** or 4START 8 ***(4NT)*** or other startup file, and the second in TCEXIT 8 ***(TC)*** or 4EXIT 8 ***(4NT)***:

In TCSTART / 4START:

```
if isfile c:\dirhist dirhistory /r c:\dirhist
```

In TCEXIT / 4EXIT:

```
dirhistory > c:\dirhist
```

This restores the previous history list if one has been saved, and saves the history when the command processor exits. Note that

Options

/A Add a directory to the directory history list.

/F Erase all entries in the directory history list.

/G Switch from a local to a global directory history list.

/L Switch from a global to a local directory history list.

/N Removes duplicate entries (oldest first) from the directory history list.

/P Wait for a key after displaying each page of the list. Your options at the prompt are explained in detail under Page and File Prompts 41.

/R Read the directory history from the specified file and append it to the list currently held in memory.

7.30 DIRS

Purpose: Display the current directory stack.

Format: DIRS

See also: PUSHD 331, POPD 326, @DIRSTACK 446 and Directory Navigation 12.

Usage:

The PUSHD 331 command adds the current default drive and directory to the directory stack, a list maintained by the command processor. The POPD 326 command removes the top entry of the directory stack and makes that drive and directory the new default. The DIRS command displays the contents of the directory stack, with the most recent entries on top (*i.e.*, the next POPD will retrieve the first entry that DIRS displays).

For example, to change directories and then display the directory stack:

```
[c:\] pushd c:\database
[c:\database] pushd d:\wordp\memos
[d:\wordp\memos] dirs
c:\database
c:\
```

The directory stack holds 511 characters, enough for 20 to 40 typical drive and directory entries.

7.31 DO

Purpose: Create loops in batch files.

Format: DO loop_control
commands
[ITERATE 248]
commands
[LEAVE 248]
commands
ENDDO

Loop_control formats

DO 247 count
DO FOREVER 247
DO 247 varname = start TO end [BY step]
DO WHILE 247 condition
DO UNTIL 247 condition
DO UNTIL DATETIME 247 date time
DO FOR 247 n [SECONDS | MINUTES | HOURS]
DO 248 varname IN [range...] [/I:"text" /S[n] /A:[-|+]hsad] fileset
DO 248 varname IN [/T"delimsiters"] /L stringset
DO 248 varname IN /C stringset
DO 248 varname IN @file

count	Integer in the range [0, 2 147 483 647], or an internal variable or variable function that evaluates to such a value, specifying the number of times the loop is executed.
varname	The environment variable containing the current value of the loop index, or the current filename or string, or the current line from a file. Do not prefix the variable name with **%**.
start, end, step	Integers in the range [-2 147 483 647,2 147 483 647] or internal variables or variable functions that evaluate to such values, controlling the number of times the loop is executed.
condition	A conditional expression 53 to determine whether or not the loop should be executed
fileset	A filename or list of filenames, possibly using wildcards 19
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 22	A date, time, size or exclusion range. At most one of each, in any order.
commands	One or more commands to execute each time through the loop. If you use multiple commands, they must be separated by command separators or be placed on separate lines.
date	The loop termination date in ISO 9601 format
time	The loop termination time in 24-h ***hh:mm:ss*** format
/A: 249	(Attribute select)
/C 249	Loop through each character in expression
/I"text" 249	(match description) Description range.
/L(iteral) 249	members of ***set*** are strings, not filenames
/S 249	Perform the loop in the current directory and all its subdirectories

Supports extended wildcards 19, ranges 22, and include lists 30 for the ***set***. Use wildcards with caution on LFN volumes; see LFN File Searches 31 for details.

Usage

DO can only be used in batch files 162.

Types of DO Loops

DO can be used to create several different kinds of loops.

- DO ***count***, is a counted loop. The batch file lines between DO and ENDDO are repeated ***count*** times. The command processor does not provide the user with the count of how many times the loop has been executed, though it is possible for the user to create a such a mechanism. For example::

```
set ct=0
do 5
  beep
  set ct=%@inc[%ct]
enddo
```

- DO FOREVER creates an endless loop. You must use LEAVE 248 or GOTO 282 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 53 **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 53 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/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 267 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) 267. For example:

```
do x in *.txt
  ...
enddo
```

will execute the loop once for every *.TXT* file in the current directory; each time through the loop the variable ***x*** will be set to the name of the next file that matches the file specification. The order of matches is dependent on the file system, and is totally unrelated to any characteristics of the filenames matched.

If, between DO and ENDDO, you create a new file that could be included in the list of files, it may or may not appear in an iteration of the DO loop. Whether the new file appears depends on its physical location in the directory structure, a condition over which the command processor has no control.

To use date, time, size, description, or file exclusion ranges 22 for the ***set*** place them just before the filename(s), for example:

```
do x in /[d9-1-2004,9-31-2004] *.txt
```

To execute the loop once for each line of text in the clipboard, use **CLIP:** as the file name (*e.g.* `DO X IN @CLIP:`). **CLIP:** will not return any data unless the clipboard contains text. See Redirection 36 for more information on **CLIP:**.

- DO ***varname*** IN /L ***stringset*** executes the commands between DO and ENDDO once for every string literal in ***stringset***, setting ***varname*** to each in turn.

- DO ***varname*** IN /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 the command processor, such as redirection and piping symbols, within the file***.*** Use SETDOS 354 /X with appropriate codes as needed.

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 286 or IFF 287 command (group):

```
do while "%var" != "%val1"
 ...
 if "%var" == "%val2" leave
enddo
```

Usage Notes

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

DO loops can be nested, i.e. you can have a DO/ENDDO loop within another DO/ENDDO loop.

The DO and ENDDO commands must be on separate lines, and cannot be placed within a command group 66, or on the same line as other commands. (This is the reason DO loops cannot be used in aliases.) However, commands within the DO loop can use command groups or the command separator in the normal way. For example, the following command will not work properly, because the DO and ENDDO are inside a command group and are not on separate lines:

```
if "%a" == "%b" (do i = 1 to 10 & echo %i & enddo)
```
invalid command line

However, this batch file fragment uses multiple commands and command grouping within the DO, and will work properly:

```
do i = 1 to 10
  ...
  if "%x1" == "%x2" (echo Done! & leave)
  ...
enddo
```

You can exit from all DO / ENDDO loops by using GOTO 282 to a line past the corresponding ENDDO. However, be sure to read the cautionary notes about GOTO 282 and DO under the GOTO 282 command before using GOTO in any other way inside any DO loop.

You cannot use RETURN 340 to return from a GOSUB 280 while inside a DO loop.

Note: Do not confuse the DO command with the unrelated optional ***do*** keyword of the FOR 267 command.

Options:

/A: Select the files in a DO 248 x IN **...** by their specified attribute(s). See Attribute Switches 28 for information on the attributes which can follow **/A:**.

/C For each loop, assign the next character (including whitespace and special characters) in the expression to the DO variable.

/I"*text*" Select files in a DO 248 x IN ... by matching ***text*** in their descriptions. See Description Ranges 28 for details.

/L The parameters following DO 248 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.

/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, 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.

/T"***text***" Specify the delimiters to be used when parsing a string set.

7.32 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 251 and DRAWVLINE 252.

Usage:

DRAWBOX is useful for creating attractive screen displays in batch files.

For example, to draw a box around the edge of an 80x25 window with bright white lines on a blue background:

```
drawbox 0 0 24 79 1 bri whi on blu fill blu
```

See Colors and Color Names 552 for details about colors.

If you use ZOOM, the box appears to grow in steps to its final size. The speed of the zoom operation depends on the speed of your computer and video system.

If you use SHADOW, a drop shadow is created by changing the characters in the row under the box and the 2 columns to the right of the box to normal intensity text with a black background (this will make characters displayed in black disappear entirely).

The row and column values are zero-based, so on a standard 25 line by 80 column display, valid rows are 0 - 24 and valid columns are 0 - 79.
Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits). DRAWBOX checks for valid row and column values, and displays a "Usage" error message if any values are out of range.

(TC) The maximum ***row*** value is determined by the current height of the ***TC*** window. The maximum

column value is determined by the current virtual screen width (see Resizing the Take Command Window 84 for more information).

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 the U.S. English extended ASCII character set. If your system is configured for a different country or language, or if you use a font which does not include these line drawing characters, the box or lines may not appear on your screen as you expect.

(TC) They will only appear correctly if you have configured ***TC*** to use a font, such as Terminal or Lucida Console, which contains standard extended ASCII characters.

7.33 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 250 and DRAWVLINE 252.

Usage:

DRAWHLINE is useful for creating attractive screen displays in batch files. It detects other lines and boxes on the display, and creates the appropriate connector characters when possible (not all types of lines can be connected with the available characters).

For example, the following command draws a double line along the top row of the display with green characters on a blue background:

```
drawhline 0 0 80 2 green on blue
```

The *row* and *column* values are zero-based, so on a 25 line by 80 column display, valid *rows* are 0 - 24 and valid *columns* are 0 - 79.
Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits). If either value is out of range, DRAWHLINE displays a "Usage" error message.

(TC) The maximum *row* value is determined by the current height of the ***TC*** window. The maximum *column* value is determined by the current virtual screen width (see Resizing the Take Command

Window 84 for more information).

If ***row*** is set to 999, the line will be centered vertically. If ***column*** is set to 999, the line will be centered horizontally.

See Colors and Color Names 552 for details about colors.

DRAWHLINE uses the standard line and box drawing characters in the U.S. English extended ASCII character set. If your system is configured for a different country or language, or if you use a font which does not include these line drawing characters, the box or lines may not appear on your screen as you expect.

(TC) They will only appear correctly if you have configured ***TC*** to use a font, such as Terminal or Lucida Console, which contains standard extended ASCII characters.

7.34 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 250 and DRAWHLINE 251.

Usage:

DRAWVLINE is useful for creating attractive screen displays in batch files. It detects other lines and boxes on the display, and creates the appropriate connector characters when possible (not all types of lines can be connected with the available characters).

For example, to draw a double width line along the left margin of the display with bright red characters on a black background:

```
drawvline 0 0 25 2 bright red on black
```

The *row* and *column* values are zero-based, so on a 25 line by 80 column display, valid rows are 0 - 24 and valid columns are 0 - 79. Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits). If either value is out of range, DRAWVLINE displays a "Usage" error message.

(TC) The maximum *row* value is determined by the current height of the ***TC*** window. The maximum *column* value is determined by the current virtual screen width (see Resizing the Take Command Window 84 for more information).

See Colors and Color Names 552 for details about colors.

DRAWVLINE uses the standard line and box drawing characters in the U.S. English extended ASCII

character set. If your system is configured for a different country or language, or if you use a font which does not include these line drawing characters, the box or lines may not appear on your screen as you expect.

(TC) They will only appear correctly if you have configured ***TC*** to use a font, such as Terminal or Lucida Console, which contains standard extended ASCII characters.

7.35 ECHO

Purpose: Enable or disable batch file or command line echoing, display the echoing status on stdout 571, or display a message on stdout 571.

Format: ECHO [ON | OFF | message]

message Text to display.

See also the commands ECHOS 255, ECHOSERR 256, ECHOERR 254, SCREEN 342, SCRPUT 344, TEXT 376 and VSCRPUT 391, and the internal variable _ECHO 416.

Usage:

The ECHO command has two unrelated, independently functioning purposes:

- Command line echoing 253
- Message display 253

Command line echoing

When command line echoing is enabled, each command is displayed on stdout 571 after it is fully parsed, aliases, functions, and variables expanded, but before it is executed.

Echoing control

The command processor 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 354 command, the configuration dialogs 151, or the BatchEcho 164 directive.

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 416.

Message display

If the ECHO command has a message (the whole command tail 559, 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 571, regardless of the applicable echoing state. Any display sent to stdout 571 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 166) or precede them with the escape character 69, or use the /X option of the SETDOS 354 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 541 **NUL** character cannot be included in ***message*.**
- If stdout 571 is the console, after displaying ***message*** on the current line, the cursor will be moved to the beginning of the next line.
- If stdout 571 is a file, the **CR LF** sequence will be appended to ***message***.

To display a blank line, use one of the forms below:

echo ``	(two consecutive back quotes), or
echo.	(special syntax for compatibility with CMD.EXE).

Examples

This command will display a message:

```
echo Processing your print files...
```

The command

```
echo    This text is indented 3 spaces   ``
```

will display 3 leading and 3 trailing spaces.

7.36 ECHOERR

Purpose: Display a message to the standard error device (stderr 571).

Format: ECHOERR message

message Text to display.

See also: ECHO 253, ECHOS 255, and ECHOSERR 256.

Usage:

ECHOERR (like ECHO 253 in message display mode) parses and expands ***message***, and displays it on stderr 571 (usually the screen), instead of stdout 571. Even if stdout 571 of a batch file is redirected or piped, ECHOERR will still display a screen message, unless stderr 571 is redirected or piped (see

Redirection 35). Any display sent to stderr 571 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 166) or precede them with the escape character 69, or use the /X option of the SETDOS 354 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 541 **NUL** character cannot be included in ***message*.**
- If stderr 571 is the console, after displaying ***message*** on the current line, the cursor will be moved to the beginning of the next line.
- If stderr 571 is a file, the **CR LF** sequence will be appended to ***message***.

7.37 ECHOS

Purpose: Display a message to standard output (stdout) 571 without a trailing carriage return / line feed.

Format: ECHOS message

message Text to display.

See also: ECHO 253, ECHOERR 254, ECHOSERR 256, SCREEN 342, SCRPUT 344, TEXT 376, and VSCRPUT 391.

Usage:

ECHOS, like ECHO 253 in message display mode, parses, expands, and displays ***message*** on stdout 571. However, any display sent to stdout 571 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 166) or precede them with the escape character 69, or use the /X option of the SETDOS 354 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 541 **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.

7.38 ECHOSERR

Purpose: Display a message to the standard error device (stderr 571) without a trailing carriage return / line feed.

Format: ECHOSERR message

message Text to display.

See also: ECHO 253, ECHOS 255, and ECHOERR 254.

Usage:

ECHOSERR acts as a combination of ECHOS 255 and ECHOERR 254. It parses and expands ***message***, and displays it on stderr 571. However, any display sent to stderr 571 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 166) or precede them with the escape character 69, or use the /X option of the SETDOS 354 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 541 **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

7.39 EJECTMEDIA

Purpose: Eject removable media in the specified drive(s)

Format: EJECTMEDIA drive ...

Usage:

EJECTMEDIA will eject removable media, such as CD-ROMs, DVDs, etc. (It is not intended for unmounting USB drives.)

7.40 ENDLOCAL

Purpose: Restore the saved disk drive, directory, environment, alias list, and special characters, and exports selected variables.

Format: ENDLOCAL [/D] [exportvar …]

/D(ont restore) 258

See also: SETLOCAL 358.

Usage:

The SETLOCAL 358 command in a batch file saves the current disk drive, default directory, all environment variables, the alias list, and the command separator, escape character, parameter character, decimal separator, and thousands separator. It does not save the user-defined function list. ENDLOCAL restores everything that was saved by the previous SETLOCAL 358 command, except as described below.

For example, this batch file fragment saves everything, removes all aliases so that user aliases will not affect batch file commands, changes the disk and directory, changes the command separator, runs a program, and then restores the original values:

```
setlocal
unalias *
cdd d:\test
setdos /c~
program ~ echo Done!
endlocal
```

SETLOCAL 358 / ENDLOCAL may be nested within a single batch file up to 16 levels of nesting. You can also have multiple, separate SETLOCAL 358 / ENDLOCAL pairs within a batch file, and nested batch files can each have their own SETLOCAL 358 / ENDLOCAL.

You cannot use SETLOCAL 358 and ENDLOCAL in an alias or at the command line.

An ENDLOCAL is performed automatically at the end of a batch file if you forget to do so. If you invoke one batch file from another without using CALL 209, 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 358.

- **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 358 / 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 258 below.

Options:

/D (Don't restore directory) Export the current directory: the original drive and directory saved by SETLOCAL 358 will not be restored.

7.41 ESET

Purpose: Edit an environment variable, alias or function definition.

Format: ESET [/A /D /F /M /S /U /V /W] name

name The name of an environment variable,function or alias to edit.

/A(lias) 259
/D(efault environment) 259
/F(unction) 259
/M(aster environment variable) 259
/S(ystem variable) 259
/U(ser variable) 259
/V(olatile variable) 259
/W(indowed Editing) 259

See also: ALIAS 187, FUNCTION 275, SET 351, UNALIAS 386, UNFUNCTION 387, and UNSET 388.

Usage:

ESET allows you to edit an environment variable, alias or function definition using line editing commands (see Command Line Editing 47).

For example, to edit the executable file search path:

```
eset path
path=c:\;c:\dos;c:\util
```

To create and then edit an alias:

```
alias d = dir /d/j/p
eset d
d=dir /d/j/p
```

Unless a specific data type is specified by one of the option switches **/A**, **/D**, **/F**, **/M**, **/S**, **/U** or **/V**, ESET will search for ***name*** among environment variables first and then among aliases, thus if ***name*** is both a variable and an alias, ESET will edit the variable ***name***, and ignore the alias ***name***.

To edit variables defined in the Windows Registry or to edit functions, you **must** use the appropriate option switch.

Environment variable and alias names are limited to 80 characters. The total length of the name and value combined is limited by the maximum line length (8,191 characters). If you use special techniques to create a longer environment variable, ESET will edit it, provided it contains no more than 8,191 characters.

Note: You cannot use ESET with GOSUB variables 280.

If you have enabled global aliases (see ALIAS 187), any changes made to an alias with ESET will immediately affect all other copies of the command processor which are using the same alias list. Similarly, if you have enabled global functions (see FUNCTION 275), any changes made to a function using ESET /F will immediately affect all other copies of the command processor which are using the same function list.

Registry Variables: **Default**, **System**, **User**, and **Volatile** registry variables can be manipulated with the ESET command's **/D**, **/S**, **/U** and **/V** switches, respectively. For example, to edit volatile variable ***myvar*** from the registry, use:

```
eset /v myvar
```

Use caution when directly modifying registry variables as they may be essential to various Windows processes and applications.

Options:

/A Edit the named alias even if an environment variable of the same name exists. If you have an alias and an environment variable with the same name, you must use this switch to be able to edit the alias.

/F Edit a user-defined function.

/D Edit a "default" variable in the registry (HKU\.DEFAULT\Environment).

/M Edit the "master" environment (inherited by the command processor at startup). Note that the master environment is only used if you run START 364 with the /I option.

/S Edit a "system" variable in the registry (HKLM\System\CurrentControlSet\Control\Session Manager\Environment).

/U Edit a "user" variable in the registry (HKCU\Environment).

/V Edit a "volatile" variable in the registry (HKCU\Volatile Environment).

/W Edit the environment variable, alias, or function list in a popup window like that used by the Batch File Debugger 201. Note that any variable name passed to ESET will be ignored when this option is used. Non-environment variables (/D, /S, /U, /V) may not be edited with this option.

7.42 EVENTLOG

Purpose: Write a string to the Windows event log.

Format: EVENTLOG [/Cn /E /I /S source /W] *message*

message The text to write.
source The source for this message.

/C(ategory) 260 /S(ource) 260
/E(rror) 260 /W(arning) 260
/I(nformational) 260

See also: HISTORY 285 and LOG 303.

Usage:

EVENTLOG posts messages to the Windows application event log. Each message can be a maximum of 8,191 characters long. You cannot use the command separator 110 character ([**&**]) or the

redirection 35 symbols (**| > <**) in an EVENTLOG message, unless you enclose the message in quotes 166 or precede the special characters with the escape character 69.

By default, the text written with EVENTLOG is stored in the event log as informational messages. You can store warning and error messages by using the **/W** and **/E** switches.

Messages in the log can be reviewed with the Windows Event Log viewer.

If you do not have proper registry permissions when you execute the EVENTLOG command and/or the key cannot be created, EVENTLOG will fail and display an error. EVENTLOG is primarily intended for use by users with **Administrator** status.

Options:

/Cn Set the event category. The value can be from 0-999999; Windows defines 0-7 as:

0 - None
1 - Devices
2 - Disk
3 - Printers
4 - Services
5 - Shell
6 - System
7 - Network

/E Store the message as an error entry in the event log.

/I Store the message as an informational entry in the event log. This is the default if no switch is used.

/S Specify the event log entry source. (If the source contains white space, it must be double-quoted). For example:

```
eventlog /sCompiling  /I Your message here.
```

/W Store the message as a warning entry in the event log.

7.43 EXCEPT

Purpose: Perform a command on all available files except those specified.

Format: EXCEPT [/I"text"] [(*@file*) | (*file* ...)] *command*

file The file or files to exclude from the command.
@file A text file containing the names of the files to exclude, one per line (see @file lists 32 for details).
command The command to execute, including all appropriate parameters and switches.

/I (match description) 262

See also: ATTRIB 198 and File Exclusion Ranges 27.

File Selection

Supports extended wildcards 19, ranges 22, multiple file names 29, and include lists 30. 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 31 for details.

Usage:

EXCEPT provides a means of executing a command on a group of files and/or subdirectories, and excluding a subgroup from the operation. The ***command*** can be an internal command or alias, an external command, or a batch file.

You may use wildcards to specify the files to exclude from the ***command***. The first example erases all the files in the current directory except those beginning with *MEMO*, and those whose extension is *.WKS*. The second example copies all the files and subdirectories on drive C to drive D except those in *C:\MSC* and *C:\DOS*, using the COPY command:

```
except (memo*.* *.wks) erase *.*
except (c:\msc c:\dos) copy c:\*.* d:\ /s
```

When you use EXCEPT on an LFN drive, you must quote any file names inside the parentheses which contain white space or special characters. For example, to copy all files except those in the *"Program Files"* directory to drive E:\:

```
except ("Program Files") copy /s *.* e:\
```

EXCEPT will assume that the files to be excluded are in the current directory, unless another directory is specified explicitly.

EXCEPT prevents operations on the specified file(s) by setting the hidden attribute, performing the command, and then clearing the hidden attribute. If the command is aborted in an unusual way, you may need to use the ATTRIB command to remove the hidden attribute from the file(s). Files which already had the hidden attribute, and are included in the set matching EXCEPT, will not be hidden after EXCEPT is 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 225 /Z, and the **/H** (process hidden files) switch available in internal file processing commands.

File exclusion ranges 27 provide a faster and more flexible method of excluding files from internal commands, and do not manipulate file attributes, as EXCEPT does. However, exclusion ranges can only be used with internal commands; you must use EXCEPT for external commands.

Date, time, and size ranges can be used immediately after the word EXCEPT to further qualify which files should be excluded from the ***command***. If the ***command*** is an internal command that supports ranges, an independent range can also be used in the ***command*** itself. You can also use a file exclusion range within the EXCEPT command; however, this will select files to be excluded from EXCEPT, and therefore included in execution of the ***command***.

You can use command grouping 66 to execute multiple ***commands*** with a single EXCEPT. For example, the following command copies all files in the current directory whose extensions begin with *.DA*, except the *.DAT* files, to the *D:\SAVE* directory, then changes the first two characters of the extension of the copied files to *.SA*:

```
except (*.dat) (copy *.da* d:\save & ren *.da* *.sa*)
```

If you use filename completion (see Filename Completion 58) to enter the filenames inside the parentheses, type a space after the open parenthesis before entering a partial filename or pressing

Tab. Otherwise, the command line editor will treat the open parenthesis as the first character of the filename to be completed.

Option:

/I"text" Select files by matching text in their descriptions. The text can include wildcards 19 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 32 for details.

7.44 EXIT

Purpose: Exit the current command processor session.

Format: EXIT [/B] [*value*]

value The numeric exit code to return.

/B 262 (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.EXE**. The CANCEL 211 and QUIT 333 commands are generally more flexible for use in batch files.

7.45 FFIND

Purpose: Search for files by name or contents.

Format: FFIND [/A[[:][-]rhsadecijopt /B /C /D[*list*] /E["text"] /F /G /I /I"text" /K /L /M /N /O[[:][-]acdeginorsu] /P /R /S[n] /[T|X]"xx" /U /V /Y /+n /-n] file...

list A list of disk drive letters (without colons).
file The file, directory, or list of files or directories to display.

/A(ttribute select) 264
/B(are) 264
/C(ase sensitive) 264
/M (no footers) 265
/N(ot) 265
/O(rder) 265

File Selection

Supports extended wildcards 19, ranges 22, multiple file names 29, and include lists 30.

Internet: Can be used with FTP Servers 42.

Usage:

FFIND is a flexible search command that looks for files based on their names and their contents. Depending on the options you choose, FFIND can display filenames, matching text, or a combination of both in a variety of formats.

(TC) Most of the functions provided by FFIND are also available in the Find Files / Text dialog 80, accessible from the Utilities menu. You can use the FFIND command, the dialog, or both, depending on your needs.

If you want to search for files by name, FFIND works much like the DIR command. For example, to generate a list of all the *.BTM* files in the current directory, you could use the command

```
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 the command processor's extended wildcards 19 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 526 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"4NT" ftp://jpsoft.com/index
```

You can also use the IFTP command to start an FTP session on a server, and then use an abbreviated syntax to specify the files and directories you want. For more information, see Using FTP/HTTP Servers 42 and IFTP 288.

Note that searching for text in files on FTP servers (as in the command above) will be slow as the data from each file searched must be retrieved from the server and transferred to your computer to be checked for the search string

Options:

/A: Select only those files that have the specified attribute(s) set. See Attribute Switches 28 for information on the attributes which can follow **/A:**.

/B Display file names only and omit the text that matches the search. This option is only useful in combination with **/T** or **/X**, which normally force FFIND to display file names and matching text.

/C Perform a case-sensitive search. This option is only valid with **/T**, which defaults to a case-insensitive search. It is not needed with a **/X** hexadecimal search, which is always case-sensitive.

/D Search all files on one or more drives. If you use **/D** without a list of drives, FFIND will search the drives specified in the list of files. If no drive letters are listed, FFIND will search all of the current drive. You can include a list of drives or a range of drives to search as part of the **/D** option. For example, to search drives C:, D:, E:, and G:, you can use either of these commands:

```
ffind /dcdeg ...
ffind /dc-eg ...
```

Drive letters listed after **/D** will be ignored when processing file names which also include a drive letter. For example, this command displays all the *.BTM* files on C: and

E:, but only the *.BAT* files on D:

```
ffind /s /dce *.btm d:\*.bat
```

/E Display filenames in upper case.

/E"text" Search for a regular expression 526. The regular expression must be contained in double quotes if it contains spaces, punctuation, or wildcard characters. See also **/T**.

/F Stops the search after the first match.

/G Change to the directory where the match was found (must be used with **/F**).

/I Only meaningful when used in conjunction with the **/T "text"** option. Suppresses the recognition of wildcard characters in the search text. This option is useful if you need to search for characters that would normally be interpreted as wildcards: *****, **?**, **[**, and **]**.

/I"text" Select filenames by matching text in their descriptions. The text can include wildcards 19 and extended wildcards. The search text must be enclosed in double quotes, and must follow the **/I** immediately, with no intervening spaces. You can select all filenames that have a description with **/I"[?]*"**, or all filenames that do not have a description with **/I"[]"**.

/K Suppress the display of the header or filename for each matching text line.

/L Include the line number for each text line displayed. FFIND numbers lines beginning with **1**, unless ListRowStart 129 is set to **0** in the INI file 91. 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 71, whichever comes first.

/M Suppress the footer (the number of files and number of matches) at the end of FFIND's display.

/N Reverse the meaning of the search, i.e., report only files which contain no match. Setting **/N** will also set **/B**, i.e. searches are on a file-by-file basis; FFIND cannot search for all lines without match.

/O Set the sort order for the files that FFIND displays. You may use any combination of the following sorting options; if multiple options are used, the listing will be sorted with the first sort option as the primary key, the next as the secondary key, and so on:

- **-** Reverse the sort order for the next option
- **a** Sort names and extensions in standard ASCII order, rather than sorting numerically when digits are included in the name or extension
- **c** Sort by compression ratio (the least compressed file in the list will be displayed first)
- **d** Sort by date and time (oldest first); for drives which support long file names
- **e** Sort by extension
- **g** Group subdirectories first, then files
- **i** Sort by file description (ignored if **/O:c** is also used)
- **n** Sort by filename (this is the default)
- **o** Sort by owner
- **r** Reverse the sort order for all options
- **s** Sort by size

u Unsorted

/P Wait for a key to be pressed after each screen page before continuing the display. Your options at the prompt are explained in detail under Page and File Prompts 41.

/R Only meaningful when used in conjunction with the **/T "text"** or **/X** options. Searches each file from the end backwards to the beginning. This option is useful if you want to display the last occurrence of the search string in each file instead of the first (the default). It may also speed up searches for information that is normally at the end of a file, such as a signature.

/S Display matches from the current directory and all of its subdirectories. By default, FFIND processes only those subdirectories without the Hidden or System attributes. To view hidden or system subdirectories use **/A** along with **/S**. If you specify a number following the **/S,** FFIND will limit the subdirectory recursion to the number specified. For example, if you have a directory tree "\a\b\c\d\e", /S2 will only go to the "a", "b", and "c" directories.

/T"text" Specify the text search string. **/T** must be followed by a text string in double quotes (*e.g.*, /t"color"). FFIND will perform a case-insensitive search unless you also use the **/C** option. For a hexadecimal search and/or hexadecimal display of the location where the search string is found, see **/X**. You can specify a search string with either **/T** or **/X**, but not both.

/U Display only the summary.

/V Show every matching line. FFIND's default behavior is to show only the first matching line, then to the next file. This option is only valid with **/T** or **/X**.

/X["xx.."] Specify hexadecimal display and an optional hexadecimal search string.

If **/X** is followed by one or more pairs of hexadecimal digits in quotes (*e.g.*, /x"44 63 65"), FFIND will search for that exact sequence of characters or data bytes without regard to the meaning of those bytes as text. If those bytes are found, the offset is displayed (in both decimal and hexadecimal). A search of this type will always be case-sensitive.

If **/X** is **not** followed by a hexadecimal search string it must be used in conjunction with **/T**, and will change the output format to display offsets (in both decimal and hexadecimal) rather than actual text lines when the search string is found. For example, this command uses **/T** to display the first line in each BTM file containing the word "hello":

```
ffind /t"hello" *.btm
—- c:\test.btm:
echo hello

1 line in 1 file
```

If you use the same command with **/X**, the offset is displayed instead of the text:

```
ffind /t"hello" /x *.btm
—- c:\test.btm:
Offset: 1A

1 line in 1 file
```

You can specify a search string with either **/T** or **/X**, but not both.

/Y Prompt to stop searching after each match. This option is most useful when you are using FFIND to search for one specific file, and don't want to display all files which include a particular search string.

/[+|-]n **"/+n"** causes FFIND to skip the first **n** matches. **"/-n"** causes FFIND to stop after **n** matches.

7.46 FOR

Purpose: Repeat a command for several values of a variable.

Format: File and string mode
FOR [range...] [/I"text"] [/A:[[-|+]rhsadecijopt /D /F ["options"] /H /R [path] [/T"delimiters"]]
%var IN ([@]set) DO command | (command ... [LEAVEFOR])

Counted mode
FOR /L %var IN (start, step, end) DO command | (command ... [LEAVEFOR])

options	Parsing options for a "file parsing" FOR.
range	One or more range 22 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: 273 (Attribute select)
/D 273 (irectories only)
/F 273 (ile parsing)
/H 273 (ide dots)
/I 273 description range
/L 274 (counted loop)
/R 274 (ecursive)
/T 274 (delimiter list)

File Selection

Supports attribute switches 28, extended wildcards 19, ranges 22, multiple file names 29, and include lists 30.

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 31 for details.

Usage:

FOR begins by creating a ***set***. It then set the executes a command for every member of ***set***. The command can be an internal command, an alias, an external command, or a batch file. The members of ***set*** can be a list of file names, text strings, a group of numeric values, or text read from a list of files.

When ***set*** is made up of text or several separate file names (not an include list), the elements must be separated by spaces, tabs, or commas.

FOR includes a large number of options, some of which duplicate functions available in other internal commands. It also supports additional conventions not found in our other commands, included for compatibility with ***CMD.EXE***.

The first three sections below (Working with Files 268, Working with Text 269, and Retrieving Text from Files 269) describe the FOR command and the enhancements to it which are included in the command processor. The sections on Parsing Text from Files 269 and Counted FOR Loops 270 describe features added for compatibility with ***CMD.EXE***. The sections Directory Recursion 271 and Output Redirection 271 warn of special considerations. The section entitled Other Notes 271 contains information you may need if you use any aspect of the FOR command extensively.

Working with Files

Normally, ***set*** is a list of files specified with wildcards. For example, if you use this line in a batch file:

```
for %x in (*.txt) list %x
```

Then LIST 298 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 19, as well as extended parent directory 62 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 command processor internal commands, or other commands which require quoting filenames with white space. FOR does not quote returned names automatically, even if you included quotes in set.

If set includes filenames, the file list can be further refined by using date, time, size, description and file exclusion ranges 22. The range or ranges must be placed immediately after the word FOR. Ranges affect only those members of set which contain wildcards. For example, the FOR below will process all of the **.TXT* files that were created or updated on December 4, 2005, and of the file ABC.LST regardless of its timestamp:

```
for /[d12-4-2005,+0] %x in (*.txt abc.lst) ...
```

If ***command*** is an internal command that supports ranges, an independent range can also be used in ***command*** itself.

You can also refine the list by limiting it with the /A: 273 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 the command processor has no control. Therefore, in order to achieve consistent results you should construct FOR commands which do not create files that could become part of 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 395 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 32. 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 the command processor, for example, one of the characters <"[]|>, it may have undesirable effects. You may use the /X option of SETDOS 354 to mitigate them.

If you use **@CON** as the filename, FOR will read from standard input (typically a redirected input file) or from a pipe. If you use **@CLIP:** as the filename, FOR will read any text available from the Windows clipboard. See Redirection and Piping 35 for more information on these features.

See @file list 32 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 273 option and include one or more file names in set. When you use this form of FOR, the variable name must be a single letter, for example, **%a**.

This method of parsing, included for compatibility with ***CMD.EXE***, can be cumbersome and inflexible. For a more powerful method, use FOR with @filename 458 as the ***set*** to retrieve each line from the file, as described in the previous section, and use variable functions like @FIELD 456, @INSTR 473, @LEFT 476, @RIGHT 487, and @WORD 501 to parse the line (see Variable Functions 424 for information on variable functions).

By default, FOR will extract the first word or token from each line and return it in the variable. For example, to display the first word on each line in the file *FLIST.TXT*:

```
for /f %a in (flist.txt) echo %a
```

You can control the way FOR /F parses each line by specifying one or more parsing options in a quoted string immediately after the **/F**. The available options are:

skip=*n*: FOR /F will skip ***n*** lines at the beginning of each file before parsing the remainder of the file.

tokens=*n*, *m*, ...: By default, FOR /F returns just the first word or ***token*** from each parsed line in the variable you named. You can have it return more than one token in the variable, or return tokens in several variables, with this option.

This option is followed by a list of numbers separated by commas. The first number tells FOR /F which token to return in the first variable, the second number tells it which to return in the second variable, etc. The variables follow each other alphabetically starting with the variable you name on the FOR command line. This example returns the first word of each line in each text file in ***%d***, the second in ***%e***, and the third in ***%f***:

```
for /f "tokens=1,2,3" %d in (*.txt) ...
```

You can also indicate a range of tokens by separating the numbers with a hyphen **-**.

eol=*c*: If FOR /F finds the character ***c*** in the line, it will assume that the character and any text following it are part of a comment and ignore the rest of the line.

delims=*xxx..*: By default, FOR /F sees spaces and tabs as word or token delimiters. This option replaces those delimiters with all of the characters following the equal sign to the end of the string. This option must therefore be the last one used in the quoted options string.

usebackq : Duplicates the awkward ***CMD.EXE*** syntax. A back quoted string is executed as a command; a single quoted string is a literal string; and double quotes quote filenames in the file set. We don't recommend **usebackq** for batch files written for **4NT** or **Take Command**, as they have much more elegant ways of doing the same things.

You can also use FOR /F to parse a single string instead of each line of a file by using the string, in quotes, as ***set.*** For example, this command will assign variable ***A*** to the string ***this***, ***B*** to ***is***, etc., then display ***this***:

```
for /f "tokens=1,2,3,4" %a in ("this is a test") echo %a
```

"Counted" FOR Loop

The "counted FOR" loop is included for compatibility with ***CMD.EXE***. In most cases, you will find the DO 246 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 274 option and then include three values, separated by commas, in ***set***. These are the ***start***, ***step***, and ***end*** values. During the first iteration of the FOR loop, the variable is set equal to the ***start*** value. Before each iteration, the variable is increased by the ***step*** value. The loop ends when the variable exceeds the ***end*** value. This example will print the numbers from 1 to 10:

```
for /l %val in (1,1,10) echo %val
```

This example will print the odd numbers from **1** to **10** (**1**, **3**, **5**, **7**, and **9**):

```
for /l %val in (1,2,10) echo %val
```

The ***step*** value can be negative. If it is, the loop will end when the variable is less than the ***end*** value.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

WARNING! You must not have white space between ***start*** and the subsequent comma, nor between ***step*** and its subsequent comma. White space after the comma is accepted.

Directory Recursion

By default, FOR works only with files in the current directory or a specified directory. Option switch /R 274 specifies that the search should recursively process subdirectories. If you specify a directory name immediately after /R 274, FOR will start in that directory and then search each of its subdirectories. If no directory is specified after the /R, the search starts in the current default directory. If you do specify a directory, *and* its name includes any special characters, it must be enclosed in double quotes. For example, it must be quoted if it is specified with the aid of an environment variable, e.g., ***%windir\command*** .

There are two differences in the invocation of ***command*** caused by directory recursion:

- The loop control variable contains the full name of the matching file
- ***command*** is executed with the default directory set to the directory in which the file was found

This example processes all ***.****TXT* files in the current directory and its subdirectories:

```
for /r %x in (*.txt) ...
```

This example works with all of the ***.****BAK* files on drive ***D:***

```
for /r d:\ %x in (*.bak) ...
```

Output Redirection

The default output redirection (i.e., **for ... > filename**) creates a new output file in each iteration. If ***filename*** does not include an absolute file path, it will be created relative to the then current default directory. If you use directory recursion, this path will change for each directory processed. The simplest way to force a single target file is to enclose the whole command in parentheses, e.g.,:

```
(for %x in (set) command) > filename
```

Other Notes

- You can use either **%** or **%%** in front of the variable name (***var***) in the command. Either form will work, whether the FOR command is typed from the command line or is part of an alias or batch file. (***CMD.EXE*** which requires a single **%** if FOR is used at the command line, but requires %% if FOR is used in a batch file.) Note that you must have at least one **%** sign present.
- The variable name can be up to 80 characters long.
- If the FOR command is an alias, e.g., **alias for=*for /h**, range 22 specifications will be ignored.
- The word DO is unnecessary but accepted. Do not confuse it with the completely unrelated DO 246 command.

- If the name of the FOR variable ***var*** is a single character, for compatibility with ***CMD.EXE***, it is created in the environment in a special way that does not overwrite an existing environment variable with the same name. Wherever ***command*** contains the % sign immediately followed by the character which is the name of the FOR variable, it is replaced by its value, regardless of any characters following it. For example, the following command tries to add ***a:*** and ***b:*** to the end of PATH 399, 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 351, ESET 258, or UNSET 388 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 399 setting, then erase the PATH 399 variable completely when FOR is done.

- ***Command*** may also use the FOR variable with the special syntax of ***CMD.EXE*** described in Special syntax for CMD.EXE compatibility 164.

- 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 225 and one for %@NAME 481:

```
for %file in (*.txt) del "%@name["%file"].bak"
```

- You can use command grouping 66 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 280 to execute a subroutine for every element in ***set***. Within the subroutine, the FOR variable can be used just like environment variable. This is a convenient way to execute a complex sequence of commands for every element in ***set*** without CALL 209 ing another batch file.

- One unusual use of FOR is to execute a collection of batch files or other commands with the same parameter. For example, you might want to have three batch files all operate on the same data file. The FOR command could look like this:

  ```
  for %cmd in (filetest fileform fileprnt) %cmd datafile
  ```

  This line will expand to three separate commands:

  ```
  filetest datafile
  fileform datafile
  fileprnt datafile
  ```

- FOR statements can be nested.

LEAVEFOR

The special keyword LEAVEFOR can be used inside a FOR command group. LEAVEFOR terminates the current FOR processing and continues with the line following the FOR command, in a manner similar to that of the LEAVE keyword in a DO 246 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 28 for information on the attributes which can follow **`/A:`**.

For example, to process only those files with the archive attribute set:

```
for /a:a %f in (*.*) echo %f needs a backup!
```

Default: **/A:-D-H-S**, i.e. include only ***files*** without the ***hidden*** and ***system*** attributes.

/D Only return subdirectories, excluding "**.**" and "**..**" .

/F Return one or more words or tokens from each line of each file in ***set***. The **/F** option can be followed by one or more options in a quoted string which control how the parsing is performed. See Parsing Text From Files 269.

/H Suppresses the assignment of the "." and ".." directories to the FOR variable when directories are explicitly included using the /A: 273 option.

/I"*text*" Select filenames by matching text in their descriptions. See Description Ranges 28.

/L Interpret the three values in ***set*** as the ***start***, ***step***, and ***end*** values of a counted loop. See Counted FOR Loops 270

/R [*path*] Look in the current directory and all of its subdirectories for files in ***set***. If the **/R** is followed by a directory name, look for files in that directory and all of its subdirectories. **Warning:** if the directory name includes special characters, including "%" to indicate an environment variable, it must be enclosed in double quotes (").

/T"*text*" Specify the delimiters to be used when parsing a string set.

7.47 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 306.

Usage:

A colon [**:**] is required after each drive letter. This example displays the status of drives A and C:

```
free a: c:
```

If the volume serial number is available, it will appear after the drive label or name.

FREE supports OpenAFS 46 names.

7.48 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 275(ause) /R 275(ead from file)

See also: ASSOC 197, and Executable Extensions 33.

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 535. 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 41.

/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 571. The format of the file is the same as that of the FTYPE display.

7.49 FUNCTION

Purpose: Create, modify or display user-defined variable functions

Format: Display mode:
FUNCTION [/G /L /P] [wildname]

Direct definition mode:
FUNCTION name[=]definition

Definition file mode:
FUNCTION /R `[file...]`

file	One or more input files to read function definitions from.
wildname	Name of function whose definition is to be displayed (may contain ***** and **?** wildcards)
name	The name of the function you want to define.
definition	The value or definition of what the function should return.

/G 278 (lobal) /P 278 (ause)
/L 277 (ocal) /R 278 (ead file)

See also: UNFUNCTION 387 and ESET 258.

Usage:

Overview

FUNCTION allows you to create or display user-defined variable functions that can be used anywhere Variable Functions 424 can be used. User-defined functions are powerful alternatives to subroutines 280.

Displaying Functions

If you invoke the FUNCTION command with no parameters, it will display the current function list:

```
function
```

If you include a ***wildname***, which may include wildcards (***** or **?**), with no equal sign and no ***definition***, FUNCTION will display the current values, if any, of all functions matching ***wildname***, .e.g.:

```
function *dx*
```

will display all functions which contain ***dx*** in their name.

You can use the /P 278 option to control display scrolling when displaying functions.

Defining Functions

If you include the equal sign and ***definition***, FUNCTION will create or update the function referred to by ***name***. Any previous ***definition*** associated with ***name*** is discarded. Instead of the **=** sign, you may use one or more spaces or tab characters to separate ***name*** and ***definition***.

Once a function is defined, the definition may be edited using ESET 258 /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$** has a special meaning. The command processor interprets it to mean "the entire command line, from parameter ***n*** to the end." If ***n*** is not specified, it has a default value of **1**, so **%$**

means "the entire command line after the alias name."

The parameter **%-n$** means "the command line from parameter 1 to ***n*** - 1".

The special variable reference **%#** expands to the number of parameters passed to the function.

A function definition need not reference any parameters at all. For example:

```
function tomorrow=`%@makedate[%@inc[%@date[%_date]]]`
```

could be simply invoked as **%@tomorrow[]**.

To use the function ***name*** you invoke is as ***%@name[parameters]***,where you must specify enough parameters to assign a value to the highest numbered parameter ***referenced*** in the function definition. It may have more parameters, which will be silently ignored.

The Colors, Color Names and Codes 552 topic shows a simple example of the use of a function in a batch file.

(TC) You can also define or modify user functions with the Functions Window 82. All of the information in this section also applies to functions defined via the dialog, unless otherwise noted.

Deleting Functions

The normal method is to use the UNFUNCTION 387 command. However, it is also possible to delete a function by redefining it without a ***definition***, e.g., the command

```
function fs=
```

deletes the function ***fs***.

Local and Global Functions

Functions can be stored in either a local or global list.

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

With a global function list, all copies of the command processor will share the same function list, and any changes made to the functions in one copy will affect all other copies. This is the default in ***4NT*** and ***TC***.

You can control the type of function list from the configuration dialogs 151, with the LocalFunction 131 directive in the .INI file 91, with the **/L** and **/LF** options of the START 364 command, and with the **/L** and **/LF** startup options 4.

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

Whenever you start a second copy of the command processor which uses a local function list, it inherits a copy of the functions from the previous shell. However, any changes to the functions made in the secondary shell will affect only that shell. If you want changes made in a secondary shell to affect the previous shell, use a global function list in both shells.

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 the command processor starts, you can edit them with a text editor, reload the edited file with FUNCTION /R, and know that the same function list will be loaded the next time you start the command processor.

When you define functions in a file that will be 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 278 option), variables and functions not protected by back quotes or doubled % signs are immediately evaluated, and the result becomes part of the function definition.

Syntax errors in a function definition are not detected until it is used.

Options:

/G Switch from a local to a global function list.

/L Switch from a global to a local function list.

/P Wait for a key to be pressed after each screen page before continuing the display.

/R This option loads a list of functions from a file. If no filename is specified and input is redirected, /R will read from stdin 571. The format of the file is the same as that of the FUNCTION display:

name=definition

where ***name*** is the name of the function and ***definition*** specifies how to determine its value. You may use the equal sign **=** or whitespace to separate ***name*** and ***definition***. Back-quotes are not required.

You can add comments to the file by starting each comment line with a colon **:**.

You can load multiple files with one FUNCTION /R command by placing the names on the command line, separated by spaces:

```
function /r func1.lst func2.lst
```

Each definition in a FUNCTION /R file can be up to 8,191 characters long. The definitions can span multiple lines in the file if each line, except the last, is terminated with an

Escape Character 69.

If there is no filename parameter and input is redirected, FUNCTION /R will read from stdin 571.

7.50 GLOBAL

Purpose: Execute a command in the current directory and its subdirectories.

Format: GLOBAL [/H /I /J /N /P /Q /Sn] command

command The command to execute, including parameters and switches.

/H 280(idden directories)
/I 280(gnore exit codes)
/J 280 (only junctions)
/N 280(o junctions)
/P 280(rompt)
/Q 280(uiet)
/S 280(ubdirectory depth)

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[] 492, @full[] 466, 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 216 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 280 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 66 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 the command processor 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 41.

/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.

7.51 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 209, GOTO 282, and RETURN 340.

Usage:

GOSUB can only be used in batch files.

The command processor allows subroutines in batch files. A subroutine must start with a ***label*** (a colon [**:**] followed by a label name) which appears on a line by itself. Case differences are ignored when matching labels. The subroutine must end with a RETURN 340 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 340 call.

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 351, ESET 258, or UNSET 388 commands, and cannot be used with the DEFINED test of IF 286, IFF 287, or @IF 469.

You cannot use SET 351 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.

You cannot RETURN 340 from a GOSUB while inside a DO 246 loop.

If the command processor reaches the end of the batch file while inside a subroutine, it will automatically return to the command after the GOSUB, just as if an explicit RETURN 340 command had been included as the last line of the file.

Subroutines can be nested.

See also: user-defined functions 275.

7.52 GOTO

Purpose: Branch to a specified line inside the current batch file.

Format: GOTO [/I] label

label The batch file label to branch to.

/I(FF and DO continue) 282

See also: GOSUB 280, CALL 209.

Usage:

GOTO can only be used in batch files.

After a GOTO command in a batch file, the next line to be executed will be the one immediately following the ***label***. The ***label*** must begin with a colon [**:**] and appear on a line by itself. The colon is required on the line where the ***label*** is defined, but is not required in the GOTO command itself. Case differences are ignored when matching labels.

This batch file fragment checks for the existence of the file *CONFIG.SYS*. If the file exists, the batch file jumps to C_EXISTS and copies all the files from the current directory to the root directory on A:. Otherwise, it prints an error message and exits.

```
if exist config.sys goto C_EXISTS
echo CONFIG.SYS doesn't exist - quitting.
quit
:C_EXISTS
copy *.* a:\
```

GOTO begins its search for the ***label*** on the line of the batch file immediately after the GOTO command. If the ***label*** is not found between that position and the end of the file, GOTO will restart the search at the beginning of the file. If the label is still not found, the batch file is terminated with the error message "Label not found."

To avoid errors in the processing of nested statements and loops, GOTO cancels all active IFF 287 statements and DO 246 / ENDDO loops unless you use **/I**. This means that a normal GOTO (without **/I**) may not branch to any label that is between an IFF and the corresponding ENDIFF or between a DO and the corresponding ENDDO.

For compatibility with **CMD.EXE**, the command

```
GOTO :EOF
```

will end processing of the current batch file if the label :EOF does not exist. However, this is less efficient than using the QUIT 333 or CANCEL 211 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 187 alias) on a subsequent ENDDO, ELSE, ELSEIFF, or ENDIFF statement.

7.53 HEAD

Purpose: Display the beginning of the specified file(s).

Format: HEAD [/A:[[-][+]rhsadecijopt] /Cn /I"text" /Nn /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 32 for details).

/A: (Attribute select) 284
/C (number of bytes) 284
/I"text" (match description) 284
/N(umber of lines) 284
/P(ause) 284
/Q(uiet) 284
/V(erbose) 284

See also: LIST 298, TAIL 371, and TYPE 384.

File Selection

Supports attribute switches 28, extended wildcards 19, ranges 22, multiple file names 29, and include lists 30.

Internet: Can be used with FTP/HTTP Servers 42, e.g.

```
head "http://jpsoft.com/notfound.htm"
```

Usage:

The HEAD command displays the first part of a file or files. It is normally only useful for displaying ASCII text files (i.e. alphanumeric characters arranged in lines separated by CR/LF). Executable files (*.COM* and *.EXE*) and many data files may be unreadable when displayed with HEAD because they include non-alphanumeric characters or unusual line separators.

You can press **Ctrl-S** to pause HEAD's display and then any key to continue.

The following example displays the first 15 lines of the files *MEMO1* and *MEMO2*:

```
head /n15 memo1 memo2
```

To display text from the clipboard use **CLIP:** as the file name. CLIP: will not return any data if the clipboard does not contain text. See Highlighting and Copying Text 83 for additional information on CLIP:.

FTP Usage

HEAD can also display files on FTP/HTTP Servers 42. For example:

```
head ftp://jpsoft.com/index
```

NTFS File Streams

HEAD supports file streams on NTFS drives. You can type an individual stream by specifying the stream name, for example:

```
head streamfile:s1
```

Options:

/A: Select only those files that have the specified attribute(s) set. See Attribute Switches 28 for information on the attributes which can follow **/A:**. Do not use /A: with @file lists. See @file lists 32 for details.

/C: Display the specified number of bytes. **/C** accepts a **b**, **k**, or **m** modifiers at the end of the number. ***b*** is the number of 512-byte blocks, ***k*** is the number of kilobytes, and ***m*** the number of megabytes.

/I"text" Select files by matching text in their descriptions. The text can include wildcards and extended wildcards. The search text must be enclosed in double quotes, and must follow the **/I** immediately, with no intervening spaces. You can select all filenames that have a description with **/I"[?]*"**, or all filenames that do not have a description with **/I"[]"**. Do not use /I with @file lists. See @file lists 32 for details.

/N n Display ***n*** lines. The default is 10.

/P Pause and prompt after displaying each page.

/Q Do not display a header for each file. This is the default behavior, but an explicit /Q may be needed to override an alias that forces /V.

/V Display a header for each file.

7.54 HELP

Purpose: Display help for internal commands.

Format: HELP [topic]

topic A help topic (internal command, variable or function).

See also: The Online Help System 506.

Usage:

Online help is available for ***4NT*** and ***TC***.

The ***4NT*** and ***TC*** help system (**jphelp.chm**) uses Microsoft's HTML Help Viewer (*HH.EXE*) included in all their current Windows bundles.

If you type the command **HELP** by itself (or press F1 120 when the command line is empty), an introductory page (**Overview**) is displayed. If you type HELP plus a topic name, that topic is displayed. For example:

```
help copy
```

displays information about the COPY command and its options. All internal commands and most internal variables, functions and directives have their own topic.

You can also invoke help for the word immediately above (or immediately to the left of) the cursor by pressing Ctrl-F1 120 (this can be useful when you need the syntax for a variable function).

7.55 HISTORY

Purpose: Display or modify the history list.

Format: HISTORY [/A *command* /F /G /L /N /P /R *filename*]

command A command to be added to the history list.
filename The name of a file containing entries to be added to the history list.

/A(dd) 286
/F(ree) 286
/G(lobal) 286
/L(ocal) 286
/N(o duplicates) 286
/P(ause) 286
/R(ead) 286

See also: DIRHISTORY 244, HistoryExclude 399 and LOG 303.

Usage:

The command processor keeps a list of the commands you have entered on the command line. See Command History and Recall 49 for information on command recall, which allows you to use the history list to repeat or edit commands you have previously executed.

The HISTORY command lets you view and manipulate the command history list directly. If no parameters are entered, HISTORY will display the current command history list.

With the options explained below, you can clear the list, add new commands to the list without executing them, save the list in a file, or read a new list from a file.

The number of commands saved in the history list depends on the length of each command line. The history list size can be specified at startup from 8192 to 131071 characters (see the History 122 directive). The default size is 8192 characters.

Your history list can be stored either locally (a separate history list for each copy of the command processor) or globally (all copies of the command processor share the same list). For full details see local and global history 52.

You can use the HISTORY command as an aid in writing batch files by redirecting the HISTORY output to a file and then editing the file appropriately. However, it is easier to use the LOG /H 303 command for this purpose.

You can disable the history list or specify a minimum command line length to save with the HistMin 122 directive in the .INI file. You can prevent any command line from being saved in the history by beginning it with an "at" sign (@).

You can exclude specific commands from the History List with the HistoryExclude 399 variable.

You can control whether duplicate entries will be saved in the history list with the HistDups 121 directive in the .INI file.

You can save the history list by redirecting the output of HISTORY to a file. This example saves the command history to a file called *HISTFILE* and reads it back again immediately. If you leave out the HISTORY /F command on the second line, the contents of the file will be appended to the current history list instead of replacing it:

```
history > histfile
history /f
history /r histfile
```

If you need to save your command history at the end of each day's work, you might use the first of these commands in your *4START.BTM / TCSTART.BTM* or other startup file, and the second in *4EXIT.BTM / TCEXIT.BTM*:

```
if exist c:\histfile history /r c:\histfile
history > c:\histfile
```

This restores the previous history list if it exists, and saves the history when the command processor exits.

The command processor can also load and save the history list automatically if you use the HistFile 121 .INI directive.

Options:

/A Add a command to the history list. This performs the same function as the **Ctrl-K** key at the command line.

/F Erase all entries in the command history list.

/G Switch from a local to a global history list.

/L Switch from a global to a local history list.

/N Removes duplicate entries (oldest first) from the history list.

/P Wait for a key after displaying each page of the list. Your options at the prompt are explained in detail under Prompts 41.

/R Read the command history from the specified file and append it to the history list currently held in memory.

If you are creating a HISTORY /R file by hand, and need to create an entry that spans multiple lines in the file, you can do so by terminating each line, except the last, with an escape character 117. However, you cannot use this method to exceed the command line length limit.

7.56 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 53
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 287(gnore case)

See also: Conditional expressions 53, IFF 287, @IF 469.

Usage:

IF is most often used only aliases and batch files. It is always followed by a ***condition*** (see Conditional expressions 53), and then a ***command***. First *condition* 53 is evaluated, and if it is TRUE, ***command*** is executed. Otherwise, ***command*** is ignored.

If the condition is FALSE, **IF** returns a non-zero result, so it can be evaluated by one of the conditional expression operators (**||** or **&&**).

The **IF ... ELSE ...** syntax of CMD.EXE is also supported:

```
IF [/I] condition (command1) ELSE (command2)
```

The commands to be executed must be enclosed in parentheses (as in a command group 66). If ***condition*** is TRUE, ***command1*** is executed, if FALSE, ***command2*** is executed. **Note:** this syntax is much less powerful than the IFF 287 command, which is recommended.

Option:

/I This option is included only for compatibility with CMD.EXE. It has no effect in ***4NT*** and ***TC***, since all string comparisons are case-insensitive unless you specify a case-sensitive test (EQC).

7.57 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 53
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 286 and @IF 469.

Usage:

IFF is similar to IF 286, but it can perform one ***commandset*** when a conditional expression 53 is true and a different ***commandset*** when it is false. Repeated use of the optional ELSEIFF clause permits IFF to sequentially evaluate multiple, independent conditional expression 53s, and execute the ***commandset*** associated with the first TRUE conditional expression 53, 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 110:

- 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 110, in any combination. A ***commandset*** may also be empty, The individual commands in a ***commandset*** may include any internal command, alias, external command, or batch file.

IFF statements can be **nested,** i.e., a ***commandset*** may include another IFF / ENDIFF group. You must make sure that each individual command / ***commandset*** is syntactically correct. If an "inner" IFF / ENDIFF group is in error, it may not be detected until after the "outer" ENDIFF has been executed.

Notes

Be sure to read the cautionary notes about GOTO 282 and IFF under the GOTO 282 command before using a GOTO 282 inside an IFF statement.

If you pipe 35 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`
```

7.58 IFTP

Purpose: Open or close an FTP/FTPS session

***Format*:** IFTP [/S command /C /N /Pn /Q /V] ["ftp://[user[:password]@]server[/path]"]

user	The user name to login to the FTP site
password	The password to login to the FTP site.
server	The FTP server name.
path	The default directory on the server for this session.

/C(lose) 290 /Q(uiet) 290

/N(o paths) 290 /S(end) 290
/P(assive) 290 /V(erbose) 290

Usage:

Most file processing commands and functions in the command processor can access files on FTP servers in the same manner as files on local hard drives and a local network. Normally, each time you use the FTP feature of one of these commands or functions, it starts an FTP session, performs its task, and then closes the FTP session.

IFTP starts an FTP session which remains open until you close it or it is closed by the remote server. There are several advantages to using IFTP: the FTP connection remains open so commands execute more quickly, the syntax for accessing files on the server is shorter, and you can specify a default directory on the server for file operations.

For example, to open an FTP connection using IFTP:

```
iftp ftp://user:pwd@jpsoft.com/dir1
```

For an FTPS connection, use something like:

```
iftp ftps://user:pwd@jpsoft.com/dir1
```

This command tells IFTP to open an FTP/FTPS session with the server ***jpsoft.com,*** send ***user*** as the login username and ***password*** as the login password, and to establish the directory ***/dir1*** as the default directory for this session. The user name and password are optional; if they are not used, IFTP will attempt to log in anonymously. The double quotes are required. If you specify a password of *****, you will be prompted to enter the password (which will be appear on the screen as asterisks).

Note that in the example above ***dir1*** is a subdirectory of the FTP "root" directory – the home directory for the named FTP user. In most server configurations this is not the same as the FTP server's physical root directory.

Note: If you enter IFTP with no parameters while a connection is active, the current server name and directory will be displayed.

If you enter IFTP with only the /Q or /V switch, you change the amount of information displayed without disturbing the existing connection (if any).

Once you have established an FTP session with IFTP, you can refer to files on the server by using **ftp:** (or **ftps:**) but leaving out the user name, password, and URL of the server. On most servers, file and path names which begin **ftp:** are relative to the default directory, if any, that you specified when you opened the IFTP session; file and path names which begin **ftp:/** are relative to the root directory for the login name.

The difference can be seen in these four DIR 233 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), the command processor will not attempt to build a full directory name but will instead pass the entire string to the remote server.

You can only have one IFTP connection open at a time within a ***4NT*** or ***TC*** session. However, while you have an IFTP connection open, you can still use a complete FTP URL to perform an operation on a different server. For example, while the session above is open, you can use this command to display all files in the root directory of ***jpsoft.com***:

```
dir "ftp://jpsoft.com/*"
```

An IFTP session remains open until you explicitly close it with this command:

```
iftp /c
```

Most FTP servers "time out" after a period of inactivity. The command processor 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 418 and _iftps 418 variables.

IFTP and the other FTP features of the command processor rely on the server's compliance with Internet FTP standards. If your server is not fully compliant, or does not operate in the manner that the command processor expects, commands may not work as you intend. We urge you to test each server you use with nondestructive commands like DIR 233 before you try to copy or delete files, create or remove directories, etc.

Before you can use IFTP, you must establish the necessary connection to the Internet.

Options:

/C Use this switch, with no URL, to close an IFTP session (see the example above).

/N Pass both source and target names to the server "as is" without any attempt at expanding the paths. This option should be used with caution and only for "non standard" servers for which the default processing fails to build a suitable name.

/P /P0 disables passive mode; /P1 enables it

/Q Turn off the display of the conversation with the FTP server.

/S Allows you to send commands directly to an FTP server. The connection must have already been opened by a previous IFTP command.

/V Display the dialog with the FTP server while opening the connection. This can be useful for debugging connection problems.

See FTP Servers 42 for additional information on formatting and usage of FTP and FTPS references.

7.59 INKEY

Purpose: Get a single keystroke from the user and store it in an environment variable.

Format: INKEY [/C /D /K"keys" /P /M /Wait /X] [prompt] %%varname

prompt	Optional text that is displayed as a prompt.
varname	The variable that will hold the user's keystroke.
wait	Time to wait for a keystroke, in seconds

/C 292	Clear buffer	/P 292	Password
/D 292	Digits only	/W 292	Wait
/K 292	valid keystrokes	/X 292	no carriage return
/M 292	Mouse buttons (4NT)		

See also: INPUT 293.

Usage:

INKEY optionally displays a prompt, then it waits for a specified time (or indefinitely) for a keystroke, and places the keystroke into an environment variable. It is normally used in batch files and aliases to get a menu choice or other single-key input. Along with the INPUT 293 command, INKEY allows great flexibility in reading input from within a batch file or alias.

If ***prompt*** is included in an INKEY command, it is displayed while INKEY waits for input.

The following batch file fragment prompts for a character and stores it in the variable ***NUM***:

```
inkey /D Enter a number from 1 to 9:  %%num
```

INKEY reads standard input for the keystroke, so it will accept keystrokes from a redirected file or from KEYSTACK 296. You can supply a list of valid keystrokes with the /K 292 option.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

A standard keystroke is stored directly in the environment variable. An extended keystroke (for example, a function key or a and cursor key) is stored as a string, consisting of a leading @, followed by its scan code as a decimal number, e.g., the ***F1*** key is stored as **@59**. The **Enter** key is stored as an extended keystroke **@28.** See ASCII, Key Codes, and ANSI X3.64 Commands 540 for scan codes.

When the /M 292 option enables recognition of mouse buttons, (and /W 292 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

To test for a non-printing value returned by INKEY use the @ASCII 438 function to get the numeric value of the key, or convert the expected value of the code to a code using @CHAR 441. For example, to test for **Esc**, which has an ASCII 541 value of 27 or a left mouse button:

```
inkey Enter a key:  %%key
if "%@ascii[%key]" == "27" echo Esc pressed
if %key EQ 240 echo Left mouse button clicked
```

If you press **Ctrl-C** or **Ctrl-Break** while INKEY is waiting for a key, execution of an alias will be terminated, and execution of a batch file will be suspended while you are asked whether to cancel the batch job. A batch file can handle **Ctrl-C** and **Ctrl-Break** with the ON BREAK 315 command.

INKEY works within the command line window. If you prefer to use a dialog for user input, see the MSGBOX 313 and QUERYBOX 332 commands.

Options:

/C Clears the keyboard buffer before INKEY accepts keystrokes. If you use this option, INKEY will ignore any keystrokes which you type, either accidentally or intentionally, before it is ready to accept input.

/D Only accept numbers from **0** to **9**.

/K"*keys*" Specifies the permissible keystrokes. The list of valid keystrokes should be enclosed in double quotes. For alphabetic keys the validity test is not case sensitive. You can specify extended keys by enclosing their names in square brackets (within the quotes), for example:

```
inkey /k"ab[Ctrl-F9]" Enter A, B, Ctrl-F9 %%var
```

See Keys and Key Names 550 for a complete listing of the key names you can use within the square brackets, and a description of the key name format.

If an invalid keystroke is entered, the command processor will echo the keystroke if possible, beep, move the cursor back one character, and wait for another keystroke.

/M ***(4NT)*** Enabled only if Windows' Quick Edit is disabled (alt-space -> Properties -> Options).

/P Prevents INKEY from echoing the character.

/W Time-out period, in seconds, to wait for a response. If no keystroke is entered by the end of the time-out period, INKEY returns with the variable unchanged. This allows you to continue the batch file if the user does not respond in a given period of time. You can specify /W0 to return immediately if there are no keys waiting in the keyboard buffer. If /W is specified, mouse buttons are ignored.

For example, the following batch file fragment waits up to 10 seconds for a character, then tests to see if a "**Y**" was entered:

```
set netmon=N
inkey /K"YN" /w10 Network monitor (Y/N)?  %%netmon
iff "%netmon" == "Y" then
  rem Commands to load the monitor program
endiff
```

/X Prevents INKEY from displaying a carriage return and line feed after the user's entry.

7.60 INPUT

Purpose: Get a string from the keyboard and save it in an environment variable.

Format: INPUT [/C /D /E /Ln /N /P /Wn /X] [*prompt*] *%%varname*

prompt Optional text that is displayed as a prompt.
varname The variable that will hold the user's input.

/C(lear buffer) 293
/D(igits only) 293
/E(dit) 294
/L(ength) 294
/N(o colors) 294
/P(assword) 294
/W(ait) 294
/X (no carriage return) 294

See also: SET 351, INKEY 291, KEYSTACK 296, MSGBOX 313, and QUERYBOX 332.

Usage:

INPUT optionally displays a prompt, then waits for your entry and stores it in an environment variable. INPUT is normally used in batch files and aliases to get multi-character input (for single keystroke input, see INKEY 291).

INPUT works within the command line window. If you prefer to us a dialog for user input, see the MSGBOX 313 and QUERYBOX 332 commands.

If ***prompt*** text is included in an INPUT command, it is displayed while INPUT waits for input. Standard command line editing keys may be used to edit the input string as it is entered. If you use the **/P** password option, INPUT will echo asterisks instead of the keys you type.

All characters entered up to, but not including, the carriage return are stored in the variable.

The following batch file fragment prompts for a string and stores it in the variable FNAME:

```
input Enter the file name:  %%fname
```

INPUT reads standard input, so it will accept text from a redirected file or from the KEYSTACK 296.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

If you press Ctrl-C or Ctrl-Break while INPUT is waiting for input, execution of an alias will be terminated, and execution of a batch file will be suspended while you are asked whether to cancel the batch job. A batch file can handle Ctrl-C and Ctrl-Break itself with the ON BREAK 315 command.

You can pipe 39 text to INPUT, but it will set the variable in the "child" process used to handle the right hand side of the pipe. This variable will not be available in the original copy of the command processor used to start the pipe.

Options:

/C Discard any keystrokes pending in the keyboard buffer before INPUT begins accepting characters.

/D Only accept numbers from 0 to 9.

/E Allows you to edit an existing value. If there is no existing value for ***varname***, INPUT proceeds as if **/E** had not been used, and allows you to enter a new value.

/Ln Sets the maximum number of characters which INPUT will accept to ***n***. If you attempt to enter more than this number of characters, INPUT will beep and prevent further input (you will still be able to edit the characters typed before the limit was reached).

/N Disables the use of input colors defined in the InputColors 124 directive in the INI file 91, and forces INPUT to use the default display colors.

/P Tells INPUT to echo asterisks, instead of the characters you type.

/W Time-out period, in seconds, to wait for a response. If no keystroke is entered by the end of the time-out period, INPUT returns with the variable unchanged. This allows you to continue the batch file if the user does not respond in a given period of time. If you enter a key before the time-out period, INPUT will wait indefinitely for the remainder of the line. You can specify **/W0** to return immediately if there are no keys waiting in the keyboard buffer.

/X Prevents INPUT from adding a carriage return and line feed after the user's entry.

7.61 JABBER

Purpose: Send an IM via the JABBER network.

Format: JABBER [/S(erver) /U(ser) / P(assword)] /B target[@server] message

message The message to send

/B(uddy) 294 /S(erver) 294
/P(assword) 294 /U(sername) 294

Usage:

If /S, /U, and/or /P are not specified, JABBER will use the default values defined in the .INI file (JabberServer, JabberUser, and JabberPassword).

JABBER is intended to send single short messages on an event (for example, when a large series of file transfers is completed), not as a general replacement for an interactive IM client.

Before using JABBER, you will need to create an account on a JABBER network server. See www.jabber.org for more information on the JABBER network and for open JABBER servers.

Options:

/B Address where the message will be sent

/P Logon password on the JABBER server

/S JABBER server to log onto

/U User logon name on the JABBER server

7.62 KEYBD

Purpose: Set the state of the keyboard toggles Caps Lock, Num Lock, and Scroll Lock.

Format: KEYBD [/Cn /Nn /Sn]

n can be either **0** to turn off the toggle or **1** to turn on the toggle.

/C(aps lock) 295 /S(croll lock) 295
/N(um lock) 295

Usage:

Most keyboards have 3 toggle keys, the Caps Lock, Num Lock, and Scroll Lock. KEYBD lets you turn any toggle key on or off. It is most useful in batch files and aliases if you want the keys set a particular way before collecting input from the user.

For example, to turn off the Num Lock and Caps Lock keys, you can use this command:

```
keybd /c0 /n0
```

If you use the KEYBD command with no switches, it will display the present state of the toggle keys.

The toggle key state is typically the same for all sessions, and changes made with KEYBD in one session will therefore affect all other sessions.

Options:

/C Turn the Caps Lock key on or off.

/N Turn the Num Lock key on or off.

/S Turn the Scroll Lock key on or off.

7.63 KEYS

Purpose: Enable, disable, or display the history list.

Format: KEYS [ON | OFF | LIST]

See also: HISTORY 285.

Usage

This command is provided for compatibility with KEYS command in CMD.EXE, which controls the history list in Windows. The same functions are available by setting the HistMin 122 directive in the INI file 91, and by using the HISTORY 285 command. (CMD.EXE's KEYS command no longer has an effect, because command line editing is always enabled.)

The history list collects the commands you type for later recall, editing, and viewing. You can view the contents of the list through the history list window or by typing any of the following commands:

```
history
history /p
keys list
```

The first command displays the entire history list. The second displays the entire list and pauses at the end of each full screen. The third command produces the same output as the first, except that each line is numbered.

You can disable the collection and storage of commands in the history list by typing:

```
keys off
```

You can turn the history back on with the command:

```
keys on
```

If you issue the KEYS command without any parameters, the command processor will show you the current state of KEYS.

7.64 KEYSTACK

Purpose: Feed keystrokes to a program or command automatically.

Format: KEYSTACK [/A /R filename] [!] [/Wx] ["abc"] [keyname[n]] ...

!	Signal to clear the Keystack and the keyboard buffer.
/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.

/A(ppend) 298 /W(ait) 298
/R(ead file) 298

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 the command processor itself), you must start the program or ACTIVATE 186 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 364 command, then use the KEYSTACK command. If you start the program directly (without using START 364) 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 ExecWait 118 is set to Yes. If it is **not** set (the default), you can use KEYSTACK immediately after an application is started. However, if ExecWait 118=yes, ***4NT / TC*** 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

An exclamation mark **!** will clear all pending keystrokes in the KEYSTACK buffer.

Characters entered within double quotes (for example, "***abc"***) will be sent to the target program as is . The only items allowed outside the quotes are key names, the **!** and /W 298 options, and a repeat count.

If ***keyname*** is a single letter, it is inserted in the keystack buffer as if it had been quoted, without any spaces. For example, you could enter the string ***abc*** as **a b c**, instead of the quoted string method described above.

If ***keyname*** is a number, it is interpreted as a ***scancode***. See Key Codes and Scan Codes Table 545 for the codes you need.

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 550 for a complete listing of key names and a description of the key name and numeric key code format.

Limitations

You can store a maximum of 8,192 characters in the KEYSTACK buffer. The count is determined by the number of characters on the KEYSTACK command line, not by the actual number of characters sent to the application.

Each time the KEYSTACK command is executed, it will clear any remaining keystrokes stored by a previous KEYSTACK command.

Note

You may need to experiment with your programs and insert delays (see the /W 298 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 start ***Word***, delays about three seconds (54 clock ticks at about 1/18 second each) for ***Word*** to get started, places the keystrokes for **alt-F** (**File** pulldown menu), and 1 (open the most recently used file) into the buffer. ***Word*** receives these keystrokes and performs the appropriate actions. Notice that the two commands, START 364 and KEYSTACK are issued on a single command line. This

ensures that the keystrokes are sent to ***Word***'s window, not back to the command processor.

Option:

/A Append to the existing KEYSTACK buffer.

/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.

7.65 LIST

Purpose Display a text file, with forward and backward paging and scrolling.

Format LIST [range...] [/A:[[-|+]rhsadecijopt /B[-]n /C /Etext"/H /I /L[-]n /N /R /S /T"text" /W /X] [@file] [file...]

file A file or list of files to display.
@file A text file containing the names of the files to view, one per line (see @file lists 32 for details).
range A file selection range 22 (date 24, description 28, exclusion 27, size 24, time 26)

/A: 302 (Attribute select)
/B 302 (yte offset)
/C 302 (separate console)
/E 302 (regular expression)
/H 302 (igh bit off)
/I 302 (gnore wildcards)
/L 302 (ine offset)
/N 302 (line numbers)
/R 302 (everse)
/S 302 (tandard input)
/T 302 (search for Text)
/W 303 (rap)
/X 303 (heXadecimal display mode)

See also: HEAD 283, TAIL 371, and TYPE 384.

File Selection

Supports attribute switches 28, extended wildcards 19, ranges 22, multiple file names 29, and include lists 30.

Internet

Can be used with FTP/HTTP Servers 42.

Usage

LIST provides a fast and flexible way to view a file, without the overhead of loading and using a text editor.

For example, to display a file called *MEMO.DOC*:

```
list memo.doc
```

Note: LIST is primarily intended for displaying the contents of ASCII and Unicode text files (i.e. alphanumeric characters arranged in lines separated by CR/LF). It can be used for other files which contain non-alphabetic characters or unusual line separators, but you may need to use hexadecimal mode (see below) to display or search these files. Lines longer than 16,383 characters will be truncated unless you're in Wrap or Hex modes.

LIST displays files in the command processor window. In ***TC***, the standard tool and scroll bars are replaced with the LIST tool and scroll bars. Use the scroll bars or the cursor pad to scroll through the file. You can select the LIST command either with the mouse (on the tool bar and scrollbars) or from the keyboard. LIST recognizes the following keys and buttons:

Key (Button)	Meaning
Home	Display the first page of the file
End	Display the last page of the file
Esc (ListExit 127**)**	Exit the current file
Ctrl-C (Quit)	Quit LIST
Ctrl-PgUp	Display previous file
Ctrl-PgDn	Display next file
Up Arrow	Scroll up one line
Down Arrow	Scroll down one line
Left Arrow	Scroll left 8 columns
Right Arrow	Scroll right 8 columns
Ctrl Left Arrow	Scroll left 40 columns
Ctrl Right Arrow	Scroll right 40 columns
Del	Prompt whether to delete the file
Ins	Prompt whether to save the pipe or file to a new name
Tab	Prompt for a new default tab size
F1	Display online help
F5 (ListRefresh 129)	Refresh the display
B (ListPrevious 129**)**	Go back to the previous file in the current group of files
Ctrl-B (ListClipboard 126)	Copy the current filename to the clipboard
C (ListContinue 127**)**	Continue with the next file
e	Edit the file (with the default editor; see the Editor 116 .INI directive). If LIST is displaying a pipe, the contents are saved to the clipboard and the editor is started. (You will need to manually paste the clipboard contents.)
F (Find)	Prompt and search for a string or a sequence of hexadecimal values
Ctrl-F	Prompt and search for a string, searching backward from the end of the file
G (Goto)	Go to a specific line or, in hex mode, to a specific hexadecimal offset
H (High)	Toggle the "strip high bit" (/H 302) option
I (Info)	Display information on the current file (the full name, size, date, and time)
L(ine numbers)	Toggle the line numbering option
N (ListNext 128**)**	Find next matching string
Ctrl-N	Find previous matching string in the file
O (ListOpen 129**)**	Open a new file
Ctrl-O	Open a new file
P (Print)	Print selected pages or the entire file (make your selection in the Windows "Print" dialog)
R ListFindRegex 127**)**	Prompt and search for a regular expression
Ctrl-R	Prompt and search backwards for a regular expression
U (ListUnicode 130**)**	Toggle the Unicode display mode
W (Wrap)	Toggle the "line wrap" (/W 303) option

X (Hex) Toggle the hex-mode display (/X 303) option

Text searches performed with **F**, **N**, **Ctrl-F**, and **Ctrl-N**, or with the corresponding buttons, are not case-sensitive unless you check the Match case box in the search dialog. LIST remembers the search strings you have used in the current session; to select a previous string, use the drop-down arrow to the right of the string entry field (the **N** key and the Next button search for the top item in this drop-down list).

When the search string is found LIST displays the line containing the string at the top of the window, and highlights the string it found. Any additional occurrences of the string on the same display page are also highlighted. Highlighting is intended for use with text files. In binary files, the search string will be found but may not be highlighted properly.

If the display is currently in hexadecimal mode and you press **F** or **Ctrl-F**, you will be prompted for whether you want to search in hexadecimal mode. If so, you should then enter the search string as a sequence of 2-digit hexadecimal numbers separated by spaces, for example ***41 63 65*** (ASCII 541 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 19 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 302 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 302 to ignore wildcards in the initial search string, and /R 302 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 526 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**, unless ListRowStart 129 is set to **0** in the .INI file 91. A new line is counted for every **CR** or **LF** character (LIST determines automatically which character is used for line breaks in each file), or when line length reaches 16,383 characters, whichever comes first.

LIST normally allows long lines in the file to extend past the right edge of the screen. You can use the horizontal scrolling keys (see above) to view text that extends beyond the screen width. If you use the W command or /W 303 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 39 automatically, but if it does not, use /S 302 to explicitly specify piped input. Your ability to navigate backward through the displayed output (*e.g.* with **PgUp**)

may be limited when viewing a large amount of data through a pipe, due to the way Windows handles piped output.

To view text from the clipboard, use **CLIP:** as the file to be listed. **CLIP:** will not return any data unless the clipboard contains text. See Redirection 36 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 42 as well as the contents of HTTP/HTTPS URLs. For example:

```
list ftp://jpsoft.com/index
list http://jpsoft.com/notfound.htm
```

You can also use the IFTP 288 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 42 and IFTP 288.

- **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 526 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 571 is not redirected), LIST will open the common Windows "open file" dialog.

Most of the LIST keystrokes can be reassigned with key mapping 104 directives in the .INI file 91.

You can set the colors used by LIST with the ListColors 126 directive in the .INI file 91, or the LIST Colors selection on the Colors tab 154 of the configuration dialogs 151. If ListColors is not used, the LIST display will use the current default colors.

By default, LIST sets tab stops every 8 columns. You can change this behavior with the TabStops 146 directive.

Options

/A: Select only those files that have the specified attribute(s) set. See Attribute Switches 28 for information on the attributes which can follow /A:. Do not use **/A:** with ***@file*** lists. See @file lists 32 for details.

/B[-]n Start at byte ***n***. If ***n*** is preceded by a minus sign **-**, start ***n*** bytes from the end of the file. The /B option will only display the file from the offset to the end; you cannot go back to a point before the offset.

/C ***(4NT)*** Display the file in a separate screen buffer and restore the original buffer upon exiting LIST.

/E Search for a regular expression 526 in the first ***file***. This option is the same as pressing **R**, but it allows you to specify the search text on the command line. The regular expression must be contained in double quotes if it contains spaces, punctuation, or wildcard characters. See also **/T**.

/H Strip the high bit from each character before displaying. This is useful when displaying files created by some word processors that turn on the high bit for formatting purposes. You can toggle this option on and off from within LIST with the **H** key or the tool bar.

/I Only meaningful when used in conjunction with the /T 302 "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 302, not subsequent searches started from within LIST.

/I"text" Select files by matching text in their descriptions. See Description Ranges 28 for details.

/L[-]n Start at line ***n***. If ***n*** is preceded by a minus sign -, start ***-n*** lines from the end of the file. The **/L** option only affects the initial page display; it does not prevent you from subsequently scrolling back to the start of the file.

/N Display line numbers. You can toggle the line numbers with the **L** key.

/R Only meaningful when used in conjunction with the /T 302 "***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 233 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 19. 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 302 and /R 302.

/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.

7.66 LOADBTM

Purpose: Switch a batch file to or from BTM mode.

Format: LOADBTM [ON | OFF]

Usage:

The command processor recognizes three kinds of batch files 162: *.CMD*, *.BAT*, and *.BTM*. Batch files with a *.BTM* extension will run faster than *.BAT* or *.CMD* files, as they are loaded into memory at startup and do not open and close the batch file for each line (as do *.BAT* and *.CMD* files).

The LOADBTM command turns BTM mode on and off. It can be used to switch modes in either a *.CMD* or *.BTM* file. If you use LOADBTM with no parameter, it will display the current batch mode: LOADBTM ON or LOADBTM OFF.

Using LOADBTM to repeatedly switch modes within a batch file is not efficient. In most cases the speed gained by running some parts of the file in BTM mode will be more than offset by the speed lost through repeated loading of the file each time BTM mode is invoked.

LOADBTM can only be used within a batch file. It is most often used to convert a *.BAT* or *.CMD* file to BTM mode without changing its extension.

7.67 LOG

Purpose: Save a log of commands to a disk file.

Format: LOG [/A /E /H /W file] [ON | OFF | text]

file The name of the file to hold the log.
text An optional message that will be added to the log.

/A(ll) 305 /H 305(istory log)

/E 305 (rrors) /W 305 (rite to)

See also: HISTORY 285.

Usage:

The LOG command provides independent controls for two different methods of logging command processor activity:

- Command Log 304
- History Log 304.

Command Log

Command logging creates a record of each internal and external command executed either from the command prompt or from a batch file in the format below:

```
[date time][id]  command
```

where the ***date*** and ***time*** are formatted according to the country code set for your system, ***id*** is the process ID in hexadecimal format, and ***command*** is the actual command after any alias or variable expansion.

The LOG command controls the ***command log*** operation if it does not include the /H 305 option.

History Log

History log creates a record of each command executed from the command prompt exactly as it was entered, before aliases and variables are expanded, without any additional information.

The LOG command controls the ***history log*** operation if it includes the /H 305 option.

Logging Controls

	Command Log	*History Log*
Default file, ***4NT***	**%_boot:\4NTLOG**	**%_boot:\4NTHLOG**
Default file, ***TC***	**%_boot:\TCLOG**	**%_boot:\TCHLOG**
Directive: initial file	LogName 131	HistLogName 121
Directive: initial logging status	LogOn 131	HistLogOn 121
Directive: initial status of error logging	LogErrors 131	not applicable
Enable logging	**LOG ON**	**LOG /H ON**
Enable logging to a specific file	**LOG /W *file***	**LOG /H /W *file***
Disable logging	**LOG OFF**	**LOG /H OFF**
Display current logging status	**LOG**	**LOG /H**
Append text to log without changing logging status	**LOG *text***	**LOG /H *text***

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 file (***4NTLOG.ALL*** or ***TCLOG.ALL***). See also: the LogAll 131 directive.

/E This option saves all error messages to the ***command log*** file. See also: the LogErrors 131 directive.

/H This option makes the other options on the command line (after the **/H**) apply to the history log. For example, to turn on history logging and write to the file *C:\LOG\HLOG*:

```
log /h /w c:\log\hlog
```

/W This switch specifies a different filename for the LOG or LOG /H output. It also automatically performs a LOG ON command. For example, to turn logging on and write the log to *C:\LOG\LOGFILE*:

```
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 command processor session. Turning LOG or LOG /H off or on does not change the file name.

7.68 MD / MKDIR

Purpose: Create a subdirectory.

Format: MD [/N[et] /S] *path*...
or
MKDIR [/N[et] /S] *path*...

path The name of one or more directories to create.

/N(o update) 306 /S(ubdirectories) 306

See also: RD 333.

Internet: Can be used with FTP Servers 42.

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 522 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 14 database unless the **/N** option is specified.

You can create directories on FTP servers. For example:

```
md ftp://ftp.abc.com/data/index
```

Options:

/N If **/N** has no additional options, do not update the CD / CDD extended directory search 14 database, *JPSTREE.IDX*. This is useful when creating a temporary directory which you do not want to appear in the extended search database. **/N** takes two optional arguments:

- **e** Don't display errors. (Note that a **/Ne** alone will still update the extended directory search 14 database.)
- **t** Don't update the extended directory search 14 database. (This is the same as **/N** with no options.)

/S Allows you to create more than one directory at a time. For example, if you need to create the directory *C:\ONE\TWO\THREE* and none of the named directories exist, you can use **/S** to have MD create all of the necessary subdirectories in a single command (without the **/S**, this command will fail because the parent directory *C:\ONE\TWO* does not exist):

```
md /s \one\two\three
```

For compatibility with CMD.EXE, **/S** becomes the default if you enable command processor extensions with the **/X** switch on the command processor startup command line. See Command Line Options 4 for details on **/X**.

7.69 MEMORY

Purpose: Display the amount and status of system RAM.

Format: MEMORY

Usage:

MEMORY lists the percentage "memory load" as reported by Windows, the total and available physical RAM, the total and available page file size, the total and available virtual memory, the total and free alias and function space, and the total history space. The memory load is a figure returned by the operating system which gives an overall sense of memory utilization. It is not a precise indicator of system load or memory usage. The total page file figure shows the total number of bytes that can be stored in the file, but may not reflect the actual size of the current file on disk.

7.70 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 565) or directory (for soft link 570).
parm2 Name of the new directory entry (a file or directory reference) to be created.

/A: 308 (Attribute select)
/D 308 Delete a link

File Selection

For hard links, MKLNK supports attribute switches 28, extended wildcards 19, ranges 22, multiple file names 29, and include lists 30. 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 31 for details.

Usage:

Due to operating and file system restrictions, this command requires an NTFS volume.

The file/directory names in ***parm1*** and ***parm2*** can be fully or partially qualified, and may contain wildcards (hard links only).

If a single argument is specified and it is a junction, MKLNK will display the directory name linked to the junction.

Hard Links

If ***parm1*** is a file, and ***parm2*** does not exist, MKLNK will create a hard link 565, as described in the Glossary. If ***parm2*** exists, MKLNK reports an error.

MKLNK (and the underlying Windows API) may fail if the current directory is on a **subst** or **net use** drive, or a **UNC** volume.

Soft Links

If ***parm1*** is a directory, and ***parm2*** does not exist, MKLNK will create a soft link 570, also known as a " directory junction" or "reparse point". If ***parm2*** exists, and it is a soft link 570, 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 Unix, 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 28 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.

7.71 MOVE

Purpose: Move files to a new directory and drive.

Format: MOVE [/A:[[-]rhsadecijopt /C /D /E /G /H /I"text" /J /L /M /N[est] /O /P /Q /R /S[n] /T /U /V /W /Y /Z] [*@file*] *source*... *destination*

source	A file or list of files to move.
destination	The new location for the files.
@file	A text file containing the names of the source files to move, one per line (see @file lists 32 for details).

/A: (Attribute select) 311
/C(hanged) 311
/D(irectory) 311
/E (No error messages) 311
/G (display percent copied) 311
/H(idden and system) 311
/I"text" (match description) 311
/J (copy in restartable mode) 311
/L (ASCII FTP transfer) 311
/M(odified files) 311
/N (Disable) 312
/O (don't move if target exists) 312
/P(rompt) 312
/Q(uiet) 312
/R(eplace) 312
/S(ubdirectory tree) 312
/T(otal) 312
/U(pdate) 312
/V(erify) 312
/W(ipe) 312
/Y (force move of encrypted files) 312
/Z (overwrite) 313

Note: MOVE is a complex command. When source and destination are on the same volume and the destination doesn't exist, it's equivalent to a simple REN 337, but when the destination exists or two volumes are involved, it becomes a two-step command: a COPY 216 to the target followed, if successful, by a DEL 225 of the source. In this topic, references to "move" may apply to the entire process or only to one of the above steps specifically depending on context.

See also COPY 216, DEL 225 and RENAME 337.

File Selection

Supports attribute switches 28, extended wildcards 19, ranges 22, multiple file names 29, delayed variable expansion 31, and include lists 30. 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 31 for details.

Internet: Can be used with FTP/TFTP/HTTP/HTTPS Servers 42.

Usage:

The MOVE command moves one or more files from one directory to another, whether the directories are on the same drive or not. It has the same effect as copying the files to a new location and then deleting the originals. Like COPY 216 and RENAME 337, MOVE works with single files, multiple files, and sets of files specified with an include list.

The simplest MOVE command moves a single ***source*** file to a new location and, optionally, gives it a new name. These two examples both move one file from drive *C:* to the root directory on drive *A:*

```
[c:\] move myfile.dat a:\
[c:\] move myfile.dat a:\savefile.dat
```

In both cases, *MYFILE.DAT* is removed from drive *C:* after it has been copied to drive *A:*. If a file called *MYFILE.DAT* in the first example, or *SAVEFILE.DAT* in the second example, already existed on drive *A:*, it would be overwritten. (This demonstrates the difference between MOVE and RENAME. MOVE will move files between drives and will overwrite the destination file if it exists; RENAME will not.)

When you move a single file, the ***destination*** can be a directory name or a file name. If it is a directory name, and you add a backslash [****] to the end of the name, MOVE will display an error message if the name does not refer to an existing directory. You can use this feature to keep MOVE from treating a mistyped ***destination*** directory name as a file name, and attempting to move the ***source*** file to that name.

If you MOVE multiple files, the ***destination*** must be a directory name. MOVE will move each file into the ***destination*** directory with its original name. If the ***destination*** is not a directory, MOVE will display an error message and exit. For example, if *C:\FINANCE\MYFILES* is not a directory, this command will display an error; otherwise, the files will be moved to that directory:

```
move *.wks *.txt c:\finance\myfiles
```

The **/D** option can be used for single or multiple file moves; it checks to see whether the ***destination*** is a directory, and will prompt to see if you want to create the ***destination*** directory if it doesn't exist.

If MOVE creates one or more destination directories, they will be added automatically to the extended directory search database; see Extended Directory Searches 14 for details.

Be careful when you use MOVE with the SELECT 345 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 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 is. (These are both for compatibility with ***CMD.EXE***.)

- **FTP Usage:**

You can move files to and from Internet URLs (FTP, TFTP and HTTP). For example:

```
move ftp://ftp.abc.com/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 526 for additional details.

- **Advanced Features and Options**

MOVE first attempts to rename the file(s), which is the fastest way to move files between subdirectories on the same drive. If that fails, (*e.g.*, because the ***destination*** is on a different drive or already exists), MOVE will copy the file(s) and then delete the originals.

If MOVE must physically copy the files and delete the originals, rather than renaming them (see above), then some disk space may be freed on the ***source*** drive. The free space may be the result of moving the files to another drive, or of overwriting a larger ***destination*** file with a smaller ***source*** file. MOVE displays the amount of disk space recovered unless the ***/Q*** option is used (see below). It does so by comparing the amount of free disk space before and after the MOVE command is executed. However, this amount may be incorrect if you are using a deletion tracking system which retains deleted files for later recovery, or if another program performs a file operation while the MOVE command is executing.

When physically copying files, MOVE preserves the hidden, system, and read-only attributes of the ***source*** files, and sets the archive attribute of the ***destination*** files. However, if the files can be renamed, and no copying is required, then the *source* file attributes are not changed.

Use caution with the ***/A:*** and ***/H*** switches (both of which can allow MOVE to process hidden files) when you are physically moving files, and both the ***source*** and ***destination*** directories contain file descriptions. If the ***source*** file specification matches the description file name (normally *DESCRIPT.ION*), and you tell MOVE to process hidden files, the *DESCRIPT.ION* file itself will be moved, overwriting any existing file descriptions in the ***destination*** directory. For example, if the *C:\DATA* directory contains file descriptions, this command would overwrite any existing descriptions in the *D:\SAVE* directory:

```
[c:\data] move /h d*.* d:\save\
```

(If you remove the hidden attribute from the *DESCRIPT.ION* file the same caution applies even if you do not use ***/A:*** or ***/H***, as *DESCRIPT.ION* is then treated like any other file.)

Note: The wildcard expansion process will attempt to allow both CMD.EXE-style "extension" matching (only one extension, at the end of the word) and the advanced ***4NT***/***TC*** *string* matching (allowing things like *.*.abc) when an asterisk is encountered in the ***destination*** of a MOVE

command.

MOVE supports connected web folders. If an HTML file (i.e., with an .htm or .html extension) is copied, MOVE will look for a folder in the same directory with the same name and an extension of ".files". If it is found, the .files directory will be moved to the target directory. You can disable connected web folders by setting the registry key:

HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\NoFileFolderConnection=0

Options:

/A: Select only those files that have the specified attribute(s) set. See Attribute Switches 28 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 32 for details.

/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 32 for details.

/D Requires that the ***destination*** be a directory. If the ***destination*** does not exist, MOVE will prompt to see if you want to create it. If the ***destination*** exists as a file, MOVE will fail with an "Access denied" error. Use this option to avoid having MOVE accidentally interpret your ***destination*** name as a file name when it's really a mistyped directory name.

/E Suppress all non-fatal error messages, such as "File Not Found." Fatal error messages, such as "Drive not ready," will still be displayed. This option is most useful in batch files and aliases.

/G Displays the percentage of the file moved and the transfer rate (in Kbytes/second). This is useful when copying large files across networks or via FTP to show whether the move is proceeding.

/H Move all files, including hidden and system files. See the cautionary note under **Advanced Features and Options** above before using **/H** when both ***source*** and ***destination*** directories contain file descriptions.

/I"text" Select ***source*** files by matching text in their descriptions. The text can include wildcards. The search text must be enclosed in double quotes, and must follow the **/I** immediately, with no intervening spaces. You can select all filenames that have a description with **/I"[?]*"**, or all filenames that do not have a description with **/I"[]"**. Do not use /I with @file lists. See @file lists 32 for details.

/J Copy the file in restartable mode. The copy progress is tracked in the destination file in case the move fails. The copy can be restarted by specifying the same source and destination file names.

/L Perform FTP transfers in ASCII mode, instead of the default binary mode.

/M 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 32 for details.

/N Do everything except actually move the file(s). This option is most useful for testing what a complex MOVE command will do. **/N** displays how many files would be moved. **/N** does not prevent creation of ***destination*** subdirectories when it is used with **/S**.

A **/N** with one of the following arguments has an alternate meaning:

- **e** Don't display errors.
- **s** Don't display the summary.
- **t** Don't update the CD / CDD extended directory search 14 database *(JPSTREE.IDX)*.

/O Don't move the file(s) unless the target doesn't exist, i.e. do not overwrite an existing target..

/P Prompt the user to confirm each move. Your options at the prompt are explained in detail under Prompts 41.

/Q Don't display filenames, the total number of files moved, the percentage moved, or the amount of disk space recovered, if any. When used in combination with the **/P** option above, it will prompt for filenames but will not display the totals. This option is most often used in batch files. See also **/T**.

/R Prompt for a **Y** or **N** response before overwriting an existing ***destination*** file.

/S Move an entire subdirectory tree to another location. MOVE will attempt to create the ***destination*** directories if they don't exist, and will remove empty subdirectories after the move. When **/D** is used with **/S**, you will be prompted if the first ***destination*** directory does not exist, but subdirectories below that will be created automatically by MOVE. If MOVE /S creates one or more destination directories, they will be added automatically to the *JPSTREE.IDX* database. If you attempt to use **/S** to move a subdirectory tree into part of itself, MOVE will detect the resulting infinite loop, display an error message, and exit. Do not use /S with @file lists. See @file lists 32 for details.

If you specify a number after the /S, MOVE will limit the subdirectory recursion to that number. For example, if you have a directory tree "\a\b\c\d\e", /S2 will only affect the "a", "b", and "c" directories.

/T Don't display filenames as they are moved, but display the total number of files deleted and the amount of free disk space recovered, if any.

/U Move each ***source*** file only if it is newer than a matching ***destination*** file or if a matching ***destination*** file does not exist (also see **/C**). This option is useful for moving new or changed files from one directory to another. Do not use /U with @file lists. See @file lists 32 for details. When used with file systems that have different time resolutions (such as FAT and NTFS), **/U** will attempt to use the "coarsest" resolution of the two.

/V Verify each disk write by performing a true byte-by-byte comparison between the source and the newly-created target file. This option may significantly increase the time necessary to complete a MOVE command.

/W If the MOVE is to a different drive, after the move overwrite the source file with 0's before deleting it (like DEL /W).

/Y (XP+ Only) Force copy of an encrypted file even when the target will be decrypted (for

CMD.EXE compatibility).

/Z Overwrite read-only destination files. Without this option, MOVE will fail with an "Access denied" error if the destination file has its read-only attribute set. This option allows MOVE to overwrite read-only files without generating any errors.

7.72 MSGBOX

Purpose: Display a message box and collect the user's response.

Format: MSGBOX [/1["text"] /2["text"] /3["text"] /4["text"] /Dn /H /I /M /N /O /Px,y /Q /R /S /Tn /W] buttontype ["title"] prompt

buttontype One of **OK**, **OKCANCEL**, **YESNO**, **YESNOCANCEL**, **RETRYCANCEL**, **ABORTRETRYIGNORE**, **CANCELTRYCONTINUE**, or **CONTINUEABORT**
title Text for the title bar of the message box.
prompt Text that will appear inside the message box.

/1 (st button) 314
/2 (nd button) 314
/3 (rd button) 314
/4 (th button) 314
/D(isable temporarily) 314
/H(elp button) 314
/I(nformation icon) 314
/M (system modal) 314
/N (no sound) 315
/O (topmost window) 315
/P (screen coordinates) 315
/Q(uestion icon) 315
/R(ight justify buttons) 315
/S(top icon) 315
/T(imeout) 315
/W(arning icon) 315

See also: INKEY 291, INPUT 293, and QUERYBOX 332.

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. The command processor will automatically size and center the box on the screen. The message box has up to three response buttons (plus an optional Help button), depending on its type, as shown below.

boxtype	*button 1*	*button 2*	*button 3*
OK	OK		
OKCANCEL	OK	Cancel	
YESNO	Yes	No	
YESNOCANCEL	Yes	No	Cancel
RETRYCANCEL	Retry	Cancel	
ABORTRETRYIGNORE	Abort	Retry	Ignore
CANCELTRYCONTINUE	Cancel	Try Again	Continue
CONTINUEABORT	Continue	Abort	

If the standard message box types don't meet your needs, you can create a custom message box with up to four buttons (plus an optional Help button), specifying the text that appears on each button.

The button the user chooses is indicated using the internal variable % ? 410. Be sure to save the return value in another variable or test it immediately; because the value of % ? 410 changes with

every internal command. The following list shows the value returned for each selection:

response	% ? 410
Yes or OK	10
No	11
Cancel	12
Retry	13
Try Again	14
Continue	15
Ignore	16
Abort	17
Help	18
timeout	20
custom button 1	21
custom button 2	22
custom button 3	23
custom button 4	24

If you define custom buttons, the button type argument will be ignored.

If there is an error in the MSGBOX command itself, % ? 410 will be set as described in its documentation (see _? 410).

For example, to display a Yes or No message box and take action depending on the result, you could use commands like this:

```
msgbox yesno "Copy" Copy all files to A:?
if %_? == 10 copy *.* a:
```

MSGBOX creates a popup dialog box. If you prefer to retrieve input from the command line, see the INKEY 291 and INPUT 293 commands.

Options:

/1 If there is a text string following the option, set the custom text for the first button. Otherwise, set the first button as the default.

/2 If there is a text string following the option, set the custom text for the second button. Otherwise, set the second button as the default.

/3 If there is a text string following the option, set the custom text for the third button. Otherwise, set the third button as the default.

/4 If there is a text string following the option, set the custom text for the fourth button. Otherwise, set the fourth button as the default.

/Dn Disable the message box buttons for ***n*** seconds at startup.

/H Display a help button.

/I Display an icon consisting of a lower case "i" in a circle in the message box.

/M The message box is created as a system modal window.

/N Don't play the default sound.

/O The message box is created as a topmost window.

/Px,y The initial x,y screen coordinates. If you don't use this option, MSGBOX will center its window in the command processor window.

/Q Display a question mark icon in the message box.

/R The buttons will be right-justified (as in XP Explorer).

/S Display a stop sign icon in the message box.

/T*n* MSGBOX will wait a maximum of ***n*** seconds for a response. If the time limit expires, **%_?** will be set to 20. The time remaining before the window closes will be displayed in the default button.

/W Display an exclamation point icon in the message box.

7.73 ON

Purpose: Execute a command in a batch file when a specific condition occurs.

Format: ON BREAK [breakcommand]
ON ERROR [errorcommand]
ON ERRORLEVEL n [errorcommand]
ON ERRORMSG [errorcommand]

breakcommand command to execute when a ***break*** event occurs
errorcommand command to execute when an ***error*** event occurs

Usage:

ON sets a watch that remains in effect for the duration of the current batch file, or until replaced by another ON command. Whenever a ***break*** or ***error*** condition occurs after ON has been executed, the corresponding ***command*** is automatically executed.

***Activation of* ON BREAK**

ON BREAK will execute ***breakcommand*** if the user presses **Ctrl-C** or **Ctrl-Break**.

***Activation of* ON ERROR *and* ON ERRORMSG**

ON ERROR or ON ERRORMSG will execute ***errorcommand*** after any critical error, operating system error (such as a disk write error) or internal command error (such as a COPY 216 command that fails to copy any files, or the use of an invalid command option).

ON ERROR executes ***errorcommand*** immediately after the error occurs, without displaying any command processor error message (Windows errors may still be displayed).

ON ERRORMSG first displays the appropriate error message, then executes ***errorcommand***.

If both are specified, ON ERROR will take precedence, and ON ERRORMSG will be ignored.

Activation of ON ERRORLEVEL

ON ERRORLEVEL n will execute ***errorcommand*** when the internal ERRORLEVEL variable is greater than or equal to the integer specified by ***n.*** You can also use the IF ERRORLEVEL tests; for example:

```
ON ERRORLEVEL EQ 37 ...
```

Scope

Each time ON BREAK, ON ERROR, ON ERRORLEVEL, or ON ERRORMSG is used, it defines a new command to be executed for a break or error, respectively, and any prior command is discarded.

ON BREAK or ON ERROR[MSG] without a command restores the command processor's default handler.

ON BREAK, ON ERROR, ON ERRORLEVEL, and ON ERRORMSG only affect the current batch file. When the batch file containing ON is exited for any reason, whether temporarily (e.g., by a CALL 209 to another batch file) or permanently, the command processor's default **break** and **error** handlers become effective. A CALL 209ed 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 209 are reactivated.

Operation

The command can be any command that can be used on a batch file line by itself. Frequently, it is a GOTO 282 or GOSUB 280 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 66 as the command if you want to execute multiple commands, for example:

```
on break (echo Oops, got a break! & quit)
```

ON BREAK, ON ERROR, ON ERRORLEVEL, and ON ERRORMSG assume that you want to continue executing the batch file. After the command is executed, control automatically returns to the command in the batch file immediately after the one that was interrupted by the break or error. To avoid continuing the batch file after a break or error at the next command perform one of the following in ***command***:

- transfer control with GOTO 282,
- end the batch file with QUIT 333 or CANCEL 211
- chain to another batch file (without using CALL 209).

When handling an error condition with ON ERROR[MSG], you may find it useful to use internal

variables 402, particularly % ? 410 and % SYSERR 422, to help determine the cause of the error.

To force the command processor to ignore break or error, use the REM 336 command as your command.

Limitations

ON can only be used in batch files.

The ON ERROR[MSG] command will not be invoked if an error occurs while reading or writing redirected input, output, or a pipe.

Caution: If a break or error occurs while the command specified in ON BREAK. ON ERROR, ON ERRORLEVEL, or ON ERRORMSG is executing, the command will be restarted. This means you must use caution either to avoid or to handle any possible errors in the commands invoked by ON, since such errors can cause an infinite loop.

7.74 OPTION

Purpose: Modify or display command processor configuration.

Formats: Invoking the OPTION dialog: 317
OPTION

Temporarily changing a few options: 318
OPTION //directive=value ...

Temporarily changing a list of options: 318
OPTION @filename

Displaying the current value of an option: 318
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 91, SETDOS 354

Usage:

Invoking the OPTION Dialog

OPTION without parameters displays a property sheet which allows you to modify most of the configuration options stored in the INI file 91.

When you exit from the property sheet, you can select **Save** to save your changes in the .INI file for use in the current session and all future sessions, or select **Cancel** to discard the changes. See Configuration Dialogs 151 for more information.

In some cases, changes you make in the **Startup** section of the OPTION dialogs will only take effect when you restart your command processor. Other changes take effect as soon as you exit the dialogs with **Save** or **OK**. However, not all option changes will appear immediately, even if they have taken effect. For example, some color changes will only appear after a CLS 215 command.

OPTION handles most standard directives. More advanced directives, including all those listed under Key Mapping Directives 102 and Advanced Directives 98 cannot be modified with the OPTION dialogs. These settings must be inserted or modified in the .INI file manually.

OPTION does not preserve inline comments when saving modified settings in the .INI file. To be sure .INI file comments are preserved, put them on separate lines in the file (see .INI file 91 for details).

Setting Individual Options Temporarily

If you follow the OPTION command with one or more sequences of a double slash mark `//`, each followed by a new ***directive=value,*** the new settings will take effect immediately, and will be in effect for the current session only. This example turns off batch file echo and changes the input colors to bright cyan on black:

```
option //BatchEcho=No //InputColors=bri cya on bla
```

Option values may contain white space. However, you cannot enter an option value that contains the `//` string.

This feature is most useful for testing settings quickly, and in aliases or batch files that depend on certain options being in effect.

Changes made with `//` are temporary. They will not be saved in the .INI file.

Setting Many Options Temporarily

The command OPTION ***@filename*** allows you to temporarily modify multiple directive settings. The file specified by ***filename*** must be in the same format as an .INI file 91. 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 482 function.

7.75 OSD

Purpose: Write floating text to the display

Format: OSD [/Font=n /N /POS=top,left /RGB=r,g,b /TIME=n /TOP /BOTTOM /LEFT /RIGHT /HCENTER /VCENTER] text

/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
text	The text to display

Usage:

OSD displays text on the desktop without a surrounding window, like TV or monitor prompts.

You can combine the window positioning options. For example:

```
osd /hcenter /vcenter /n Your text here
```

7.76 PATH

Purpose: Display or alter the list of directories that the command processor will search for executable files, batch files, and files with executable extensions that are not in the current directory.

Format: PATH [directory [;directory...]]

directory The full name of a directory to include in the path setting.

See also: ESET 258 and SET 351 (the PATH command is syntactically equivalent to SET PATH).

Usage:

When the command processor is asked to execute an external command (a *.COM*, *.EXE*, *.BTM*, *.BAT*, or *.CMD* file, or an executable extension), it first looks for the file in the current directory. If it fails to find an executable file in the current directory, it will search each of the directories specified in the PATH setting.

(TC) The command processor first searches the current directory, then the ***\WINDOWS\SYSTEM32*** directory followed by the ***\WINDOWS*** directory before any directories listed in your search path. (The actual directory names may be different on your system. The command processor will determine the correct names for the "Windows" and "Windows System" directories.) These search procedures conform to the default search sequences used by Windows.

(TC) For example, after the following PATH command, ***TC*** will search for an executable file in six directories: the current directory, the two Windows directories, the root directory on drive C, then the *BIN* subdirectory on C, and then the *UTIL* subdirectory on C:

(4NT) For example, after the following PATH command, ***4NT*** will search for an executable file in four directories: the current directory, the root directory on drive C, then the *BIN* subdirectory on C, and then the *UTIL* subdirectory on C:

```
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 351, and edited with ESET 258.

The PATHEXT 399 environment variable, and the related PathExt 135 .INI directive, can be used to select the extensions to look for when searching the PATH for an executable file.

If you enter PATH with no parameters, the current path is displayed:

```
[c:\] path
PATH=C:\;C:\BIN;C:\UTIL
```

Entering PATH and a semicolon clears the search path so that only the current directory is searched for executable files. Some applications also use the PATH to search for their files.

If you include an explicit file extension on a command name (for example, WP.EXE), the search will find files with that name and extension in the current directory and every directory in the path. It will not locate other executable files with the same base name (i.e., WP.COM).

If you have an entry in the path which consists of a single period [**.**], the current directory will not be searched first, but instead will be searched when the command processor reaches the "**.**" in the path. This allows you to delay the search of the current directory for executable files and files with executable extensions. In rare cases, this feature may not be compatible with applications which use the path to find their files; if you experience a problem, you will have to remove the "." from the path while using any such application.

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.

7.77 PAUSE

Purpose: Suspend batch file or alias execution.

Format: PAUSE [text]

text The message to be displayed as a user prompt.

Usage:

A PAUSE command will suspend execution of a batch file or alias, giving you the opportunity to change disks, turn on the printer, etc.

PAUSE waits for any key to be pressed and then continues execution. You can specify the ***text*** that PAUSE displays while it waits for a keystroke, or let it use the default message:

```
Press any key when ready...
```

For example, the following batch file fragment prompts the user before erasing files:

```
pause Press Ctrl-C to abort, any other key to erase all .LST files
erase *.lst
```

If you press **Ctrl-C** or **Ctrl-Break** while PAUSE is waiting for a key, execution of an alias will be terminated, and execution of a batch file will be suspended while you are asked whether to cancel the batch job. In a batch file, you can handle **Ctrl-C** and **Ctrl-Break** yourself with the ON BREAK 315 command.

7.78 PDIR

Purpose: Display information about files and subdirectories in user-definable fields. It is a " programmable DIR" command.

Format: PDIR [ranges] [/A:[attrlist] /D /H /I"text" /K /M /N[e] /O[[:][order] /P /S[n] /T:t /(...)] [file...]

attrlist	Selection attributes (see attribute switches 28 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 323:	Attribute select	/Ne 323	No error messages
/D 323	colorize	/O 323	Order
/H 323	do not Hide . and ..	/P 324	Page pause
/I 323 "text"	description range	/S 324	Subdirectories
/K 323	show header	/T 324 [:t]	Timestamp type
/M 323	show footer	/(...) 324	output fields and format

See also: DIR 233, ATTRIB 198, DESCRIBE 230, and SELECT 345.

File Selection

Supports attribute switches 28, extended wildcards 19, ranges 22, multiple file names 29, and include lists 30.

Internet

Can be used with FTP/HTTP Servers 42.

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 233 command.

PDIR and DIR 233 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 187 command.

The /(...) option specifies which fields you want to display and how to format them. (You can have multiple /(...) options on a line.) The syntax is:

a Attributes

c Compression: Display the compression percentage on NTFS drives with compression enabled.

d[...] Date (you must specify at least one subfield, otherwise the field remains blank)

d day (2 digits, leading zero)
m month (2 digits, leading zero)
y year (4 digits)

f[...] File or Directory name (case sensitive)
P SFN path
p LFN path
N SFN filename
n LFN filename (default)

i Description

m MD5 hash value (see the @MD5 480 function)

q File or directory owner (NTFS only)

r CRC32 hash value (see the @CRC32 443 function)

s stream names (NTFS only)

sp path and stream names as pathname+filename+streamname (NTFS only)

t[...] Time (you must specify at least one subfield, otherwise the field remains blank)
h hours (2 digits, leading zero)
m minutes (2 digits, leading zero)
s seconds (2 digits, leading zero)
d milliseconds (decimal separator and 3 digits)

z[...] Size
a allocated size (this will usually be more than the physical size unless the file is compressed.)
c the size will be formatted using the thousands separator (default is a comma)
k|K|m|M|g|G|t|T (case sensitive) format as kilobytes, megabytes, gigabytes, or terabytes, as used in variable functions (see Memory Size / Disk Space / File Size Units and Report Format 425). Note that the size will be truncated, not rounded.

@function[*]
call the specified variable function 424 (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 filed will be padded with zeros instead of spaces.

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 233 command when a single filename is specified.

Most options are used to select the desired files/directories. (This is in contrast to the DIR 233 command.) The special option /(...) 324 is used to specify which characteristics of the selected files or directories should be displayed in which sequence and format.

/A:. . . Display only those files that have the specified attribute(s) set. See Attribute Switches 28 for information on the attributes which can follow **/A:.**

/D Colorize the directory listing. See DIR 233 for more information on directory colorization.

/H Show the "**.**" and "**..**" directory names (normally suppressed).

/I"*text*" Select filenames by matching text in their descriptions. See Description Ranges 28 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 default output.

- **e** Don't display errors

/O... The sorting order is applied to the listings of each subdirectory separately. Any combination of the sorting options may be used. If multiple options are specified, the listing will be sorted with the first sort option as the primary key, the next as the secondary key, and so on.

- **n** Sort by filename and extension (default). If ***e*** is also specified, sort by name only.
- **-** Reverse the sort order for the next option
- **a** Sort names and extensions in standard ASCII order, rather than sorting numerically when digits are included in the name or extension.
- **c** Sort by compression ratio (the least compressed file in the list will be displayed first).
- **d** Sort by date and time (oldest first); also see **/T:acw**
- **e** Sort by extension

g Group subdirectories first, then files
i Sort by file description (ignored if **/C** or **/O:c** is also used).
o Sort by owner
r Reverse the sort order for all options
s Sort by size
u Unsorted

/P Wait for a key to be pressed after each screen page before continuing the display. Your options at the prompt are explained in detail under Page and File Prompts 41.

/S Display file information from the current directory and all of its accessible subdirectories.

If you specify a number after the /S, PDIR will limit the subdirectory recursion to that number. For example, if you have a directory tree "\a\b\c\d\e", /S2 will only affect the "a", "b", and "c" directories.

/T:*type* Specifies which single one of the date and time fields below, available on a drive which supports long filenames, should be displayed and used for sorting:

a Last access date and time (NTFS volumes).
c Creation date and time.
w Last write date and time (default).

If /T is not specified, the default is /T:w

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 233 command.

7.79 PLAYAVI

Purpose: Play Windows .AVI (video clip) files.

Format: PLAYAVI [/A /C /S /*V*n] *filename*

filename The file to play

/A(synchronous) 325 /S(ynchronous) 325
/C(enter) 325 /V(olume) 325

Usage:

PLAYAVI "plays" an *.AVI* or Windows video clip file.

Note: This command relies on the capabilities of your Windows configurations, including access to the proper codec. See your Windows documentation for details.

By default, PLAYAVI operates in synchronous mode, which means the command processor waits for the .AVI file to complete and its window to close before continuing with the next command in a batch file or alias, or prompting you for a new command. Under ***TC***, you can change this default behavior with the **/A** switch, described below.

Options:

/A **(*TC*)** Plays the .AVI file in asynchronous mode. Control returns to the command processor prompt immediately for a new command or to execute the next command in the current batch file or alias.

/C 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 **(*TC*)** Plays the .AVI file in synchronous mode (this is the default). The command processor pauses until the file has finished playing and its window closes.

/V Sets the volume level. The range is 0 (silent) to 100.

7.80 PLAYSOUND

Purpose: Play MP3, .WAV, Midi, and other sound files.

Format: PLAYSOUND [/A /S /Van] *filename*

filename The file to play

/A(synchronous) 325 /V(volume) 325
/S(ynchronous) 325

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 the command processor waits for the sound file to complete and its window to close before continuing with the next command in a batch file or alias, or prompting you for a new command. You can change this default behavior with the /A 325 switch, described below.

You can cancel the playing of a synchronous sound file by pressing Ctrl-C or Ctrl-Break while it is playing.

Options:

/A Plays the sound file in asynchronous mode. Control returns to the command processor prompt immediately for a new command or to execute the next command in the current batch file or alias.

/S Plays the sound file in synchronous mode (this is the default). The command processor pauses until the file has finished playing and its window closes.

/V Sets the volume level. The range is 0 (silent) to 100.

7.81 PLUGIN

Purpose: Load, unload, or display current plugins

Format: PLUGIN [/B /I plugin /L plugin /P /U plugin]

/B (full pathname) 326 /P(ause) 326
/I(nfo) 326 /U(nload) 326
/L(oad) 326

Usage:

Plugins allow you to write your own internal variables, variable functions, and internal commands, put them in a DLL, and have the command processor load them at startup. Plugin names will override existing names, so you can extend and/or replace internal variables and commands. When ***4NT*** or ***TC*** start, they will automatically load any plugins in the default directory (the subdirectory PLUGINS\ in the ***4NT*** / ***TC*** installation directory). The plugins will be loaded before the startup file (4START 8 or TCSTART 8) are executed.

You can also write keystroke plugins that will be called for every keystroke entered at the command line. A keystroke plugin can perform actions when a specific key is entered, or even change the key before passing it back to the command processor.

If no options are specified, PLUGIN will display the currently loaded plugins and their internal variables, variable functions, and commands.

See the Plugin SDK for more information on developing plugins.

Options:

/B Display the full pathnames of the plugins.

/I Display information about the specified plugin, including the name, author, author's email and web addresses, description, function list, version and build numbers. The **/I** option supports wildcards.

/L Loads the specified plugin. If the filename is *****, load all plugins from the default directory (the subdirectory PLUGINS\ in the ***4NT*** / ***TC*** installation directory).

/P Wait for a key to be pressed after each screen page before continuing the display. Your options at the prompt are explained in detail under Page and File Prompts 41.

/U Unloads the specified plugin. If the filename is *****, unloads all plugins.

7.82 POPD

Purpose: Return to the disk drive and directory at the top of the directory stack..

Format: POPD [/X * n]

n The number of directories to pop

/X (exclude) 327

See also: DIRS 245, PUSHD 331, @DIRSTACK 446 and Directory Navigation 12.

Usage:

Each time you use the PUSHD 331 command, it saves the current disk drive and directory on the

internal directory stack. POPD restores the most recently saved drive and directory and removes that entry from the stack. You can use these commands together to change directories, perform some work, and return to the starting drive and directory.

Directory changes made with POPD are recorded in the directory history list and can be displayed in the directory history window 62. Read the section on Directory Navigation 12 for complete details on this and other directory navigation features.

This example saves and changes the current disk drive and directory with PUSHD 331, 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 245 command to see the complete list of saved drives and directories (the directory stack).

The POPD command followed by an asterisk [*****] clears the directory stack without changing the current drive and directory.

If the directory on the top of the stack is not on the current drive, POPD will switch to the drive and directory on the top of the stack without changing the default directory on the current drive.

Options:

/X Don't save the current directory to the Directory History list.

7.83 POSTMSG

Purpose: Post a message to a window

Format: POSTMSG "title" msg wparam lparam

title	The window title
msg	The message to send
wParam	wParam integer
lParam	lParam integer value

Usage:

POSTMSG allows you to send a Windows message to any window with a caption.

The ***title*** may contain wildcards, and POSTMSG will send the message to the first window with a matching title.

See the Windows SDK documentation for a list of possible messages and their parameters.

7.84 PRINT

Purpose: Print the specified file(s) using the application associated with each file's extension.

Format: PRINT [/A printer /D printer] filename ...

/A(dd) printer 328 /D(elete) printer 328

Usage:

Except for plain text files, Windows files cannot be printed without sending them to an associated application for interpretation and formatting. Using the extension for each file you want to print, PRINT determines if a Print action has been defined for that file type. If so, it executes the Print action and sends the file to the application for processing.

For example, if you use the command

```
print myletter.doc
```

PRINT looks up the Print command for *.DOC* files in the registry and, on most computers, will find that it is associated either with WordPad or Word. It will execute the associated program and send it the file along with the necessary command to print the file and then quit.

If PRINT cannot find a Print command for a file, it displays an error message. If there are additional files in the list you gave it to print, it will go on to the next file in the list.

PRINT depends on proper Windows File Associations 535 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.

7.85 PRIORITY

Purpose: Display or set process priority, or suspend or resume a process.

Format: PRIORITY [/Q /R /S PID | "title" ABOVE | BELOW | NORMAL | HIGH | IDLE | REALTIME]

ABOVE	Above normal priority
BELOW	Below normal priority
NORMAL	Normal (default) priority
HIGH	High priority
IDLE	Idle priority (only executes when no higher priority task is scheduled)
REALTIME	Realtime priority

/Q(uiet) 329 /S(uspend) 329
/R(esume) 329

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 (***4NT*** or ***TC***) process.

If you do not enter any arguments, PRIORITY displays all of the active processes, their current priority, the module names, and the window titles (if any).

Options:

/Q Don't display any suspend / resume messages.

/R Resume the process

/S Suspend the process

7.86 PROMPT

Purpose: Change the command line prompt.

Format: PROMPT [text]

text Text to be used as the new command line prompt.

See also: ESET 258 and SET 351 (the PROMPT command is syntactically equivalent to SET PROMPT).

Usage:

You can change and customize the command line prompt at any time. The prompt can include normal text and system information such as the current drive and directory, the time and date, and the amount of memory available. You can create an informal "Hello, Bob!" prompt or a complex prompt full of impressive information.

The prompt ***text*** can contain special commands in the form **$?**, where **?** is one of the characters listed below. Unless otherwise specified, those meta characters are case-independent.

a The ampersand character[**&**].
b The vertical bar character [**|**].
c The open parenthesis [**(**].
d Current date, in the format: **Thu 12-12-04** (the month, day, and year are formatted according to your current country settings.
D Current date, in the format: **Thu Dec 12, 2004**.
e The ASCII ESC character (decimal 27), necessary for ANSI 40 commands.
f The close parenthesis [**)**].
g The **>** character.
h Backspace over the previous character.
j Current date in ISO 9601 format (yyyy-mm-dd).
l The **<** character.
m Time in hours and minutes using 24-hour format.
M Time in hours and minutes using the default country format.
n Current drive letter.
p Current drive and directory (lower case).
P Current drive and directory (upper case on drives which do not support long filenames; directory names shown in mixed case as stored on the disk on LFN drives).
q The **=** character.
r The numeric exit code of the last external command.

s The space character.
t Current 24-hour time, in the format hh:mm:ss.
T Current 12-hour time, in the format hh:mm:ss[a|p).
u The current user.
v Windows version number, in the format **5.2**.
w Current directory, in a shortened format. If the current directory is the root or a first-level subdirectory, it is displayed as-is. If it is second level or deeper, the path is truncated (i.e., "`c:\…\config`"). (This does not work with UNC names.) $W and $w behave like $P and $p for displaying upper/lower case.
xd: Current directory on drive ***d:*** in lower case, including the drive letter (uses the actual case of the directory name as stored on the disk for LFN drives.)
Xd: Current directory on drive ***d:*** in upper case, including the drive letter.
z Current shell nesting level. The first copy of the command processor is shell level 0.
+ Display one **+** character for each directory on the PUSHD 331 directory stack.
$** The **$ character.
_ CR/LF (go to beginning of a new line).

For example, to set the prompt to the current date and time, with a ">" at the end:

```
[c:\] prompt $d $t $g
Thu  Sep 30, 2004 10:29:19 >
```

The command processor prompt can be set in 4START (4NT) or TCSTART (TC) 8 or in any batch file that runs when the command processor starts.

If you enter PROMPT with no parameters, the prompt will be reset to its default value.

You can include literal text and special characters as well as the value of any environment 395 variable, internal variable 402, or variable function 424 in a prompt. For example, if you want to include the size of the largest free memory block in the command prompt, plus the current drive and directory, you could use this command:

```
[c:\] prompt [(%%@dosmem[K]K) $p]
[(31043K) c:\data]
```

Notice that the @DOSMEM function is shown with two leading percent signs [**%**]. If you used only one percent sign, the @DOSMEM function would be expanded at once when the PROMPT command was executed, instead of every time the prompt is displayed. As a result, the amount of memory would never change from the value it had when you entered the PROMPT command. You can also use *back quotes* to delay expanding the variable function until the prompt is displayed:

```
prompt `[(%@dosmem[K]K) $p]`
```

You can use this feature along with the @EXEC 454 variable function to create a complex prompt which not only displays information but executes commands. For example, to execute an alias which checks battery status each time the prompt is displayed (enter the alias on one line):

```
alias cbatt `if %_apmlife lt 30 beep 440 4 880 4 440 4 880 4`
prompt `%@exec[@cbatt]$p$g`
```

You can include ANSI 40 escape sequences in the PROMPT by using the built-in ANSI X3.64 support in ***4NT*** and ***TC***. This example uses ANSI X3.64 sequences to set a prompt that displays the shell level, date, time and path in color on the top line of the screen (enter the command as one line):

```
prompt $e[s$e[1;1f$e[41;1;37m$e[K[$z] $d
Time: $t$h$h$h  Path: $p$e[u$e[0;32m$n$g
```

You may find it helpful to define a different prompt in secondary shells, perhaps including **$z** in the prompt to display the shell level. To do so, place a PROMPT command in your 4START 8 ***(4NT)*** or TCSTART 8 ***(TC)*** file and use IF, IFF, or SWITCH statements to set the appropriate prompt for different shells.

7.87 PUSHD

Purpose: Save the current disk drive and directory, optionally changing to a new drive and directory.

Format: PUSHD [/X path]

path The name of the new default drive and directory.

/X (exclude) 332

See also: DIRS 245, POPD 326, @DIRSTACK 446 and Directory Navigation 12.

Usage:

PUSHD saves the current drive and directory to a "last in, first out" directory stack. The POPD 326 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 245 command displays the contents of the directory stack.

To save the current drive and directory, without changing directories, use the PUSHD command by itself, with no ***path***.

If a ***path*** is specified as part of the PUSHD command, the current drive and directory are saved and PUSHD changes to the specified drive and directory. If the ***path*** includes a drive letter, PUSHD changes to the specified directory on the new drive without changing the current directory on the original drive.

This example saves the current directory and changes to C:\WORDP\MEMOS, then returns to the original directory:

```
[c:\] pushd \wordp\memos
[c:\wordp\memos] popd
[c:\]
```

When you use PUSHD to change to a directory on an LFN drive, you must quote the ***path*** name if it contains white space or special characters.

PUSHD can also change to a network drive and directory specified with a UNC name (see File Systems 521 for details).

If PUSHD cannot change to the directory you have specified it will attempt to search the CDPATH 13 and the extended directory search 14 database. You can also use wildcards 19 in the ***path*** to force an extended directory search. Read the section on Directory Navigation 12 for complete details on these and other directory navigation features.

Directory changes made with PUSHD are also recorded in the directory history list and can be displayed in the directory history window 62.

The directory stack can hold up to 511 characters, or between 20 and 40 typical entries (depending

on the length of the names). If you exceed this limit, the oldest entry is removed before adding a new entry.

Options:

/X Don't save the current directory to the Directory History list.

7.88 QUERYBOX

Purpose: Pops up a dialog box to get an input string from the user and save it in an environment variable.

Format: QUERYBOX [/D /E /Ln /P /Tn] ["title"] *prompt %%varname*

title Text for the title bar of the dialog box.
prompt Text that will appear inside the dialog box.
varname Variable name where the input will be saved.

/D(igits only) 332
/E(dit existing value) 332
/L (maximum Length) 333
/P(assword) 333
/T(imeout) 333

See also: INKEY 291, INPUT 293, and MSGBOX 313.

Usage:

QUERYBOX displays a dialog box with a prompt, an optional title, and a string input field. Then it waits for your entry, and places any characters you type into an environment variable. QUERYBOX is normally used in batch files and aliases to get text input.

QUERYBOX is similar to INPUT, except it appears as a popup dialog box. If you prefer to work within the command line window, see the INKEY and INPUT commands.

Standard command line editing keys may be used to edit the input string as it is entered. All characters entered up to, but not including, the carriage return are stored in the variable.

For example, to prompt for a string and store it in the variable NAME:

```
querybox "File Name" Enter a name:  %%name
```

If you press **Ctrl-C** or **Ctrl-Break** while QUERYBOX is waiting for input, execution of an alias will be terminated, and execution of a batch file will be suspended while you are asked whether to cancel the batch job. A batch file can handle **Ctrl-C** and **Ctrl-Break** itself with ON BREAK 315.

QUERYBOX returns a value of zero in the internal variable %_? 410 after a successful operation, and a non-zero value otherwise (timeout, cancel, etc.). Be sure to save the return value in another variable or test it immediately; because the value of **%_?** changes with every internal command.

Options:

/D Only accepts numeric values.

/E Allows you to edit an existing value. If there is no existing value for ***varname***, QUERYBOX allows you to enter a new value.

/Ln Sets the maximum number of characters which QUERYBOX will accept to ***n***.

/P Tells QUERYBOX to echo asterisks, instead of the characters you type.

/Tn Wait for a maximum of ***n*** seconds for a response.

7.89 QUIT

Purpose: Terminate the current batch file.

Format: QUIT [value]

value The numeric exit code to return to the command processor or to the previous batch file.

See also: CANCEL 211 and EXIT 262.

Usage:

QUIT provides a simple way to exit a batch file before reaching the end of the file. If you QUIT a batch file called from another batch file, you will be returned to the previous file at the line following the original CALL.

This example batch file fragment checks to see if the user entered "quit" and exits if true.

```
input  Enter your choice :  %%option
if  "%option" == "quit"  quit
```

QUIT only ends the current batch file. To end all batch file processing, use the CANCEL 211 command.

If you specify a *value*, QUIT will set the ERRORLEVEL 424 or exit code to that value. For information on exit codes see the IF 286 command, and the %? 409 variable. Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

You can also use QUIT to terminate an alias. If you QUIT an alias while inside a batch file, QUIT will end both the alias and the batch file and return you to the command prompt or to the calling batch file.

7.90 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 32 for details).

/I (match descriptions) 334
/K (no Recycle Bin) 334
/N (disable options) 334
/Q(uiet) 335
/R(ecycle bin) 335
/S(ubdirectories) 335

See also: MD 305.

File Selection

Supports extended wildcards 19, ranges 22, multiple file names 29, and include lists 30. Use wildcards with caution on LFN volumes; see LFN File Searches 31 for details.

Internet: Can be used with FTP Servers 42.

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 225 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 14 database.

You cannot remove the root directory, the current directory (**.**), any directory above the current directory in the directory tree, or any directory in use by another process. RD will delete hidden directories, for compatibility with **CMD.EXE**.

You can remove directories on FTP servers 42. For example:

```
rd ftp://ftp.abc.com/data
```

Options:

/I"text" Select directories by matching text in their descriptions. The text can include wildcards 19 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 32 for details.

/K When used with the **/S** option, this will physically delete files instead of sending them to the Windows Recycle Bin. This option overrides the default RecycleBin 139 .INI option.

/N This option takes two possible arguments:

- **e** Don't display errors.
- **t** Don't update the CD / CDD extended directory search 14 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. This option overrides the default *.INI* option.

/S This option is included only for compatibility with CMD.EXE, and should be used with **EXTREME CARE!**! It deletes all files (including hidden and system files) in the named directory and all of its subdirectories, then removes all subdirectories. **It can potentially erase all files on a drive with a single command.** You cannot use wildcards with the /S option.

Note: Do not use /S with @file lists.

7.91 REBOOT

Purpose: Reboot the computer, log off Windows, or shut down.

Format: REBOOT [/H /K /L /M[0|1] /P /R /S /V /W]

/H(ibernate 335
/K (lock) 335
/L(ogoff) 335
/M(onitor) 335
/P(ower off) 336
/R(eboot) 336
/S(hutdown) 336
/V(erify) 336
/W (standby) 336

Usage:

REBOOT will log off or shut down the operating system, or completely restart your computer. It normally performs a warm reboot, or a a shutdown and restart under Windows.

REBOOT defaults to performing a warm boot, with no prompting. The following example prompts you to verify the reboot, then does a warm boot:

```
reboot /v
```

The command processor issues the standard commands to shut down other applications and the Windows before rebooting. Windows may prompt you for additional actions, or even ignore the request altogether depending on which processes are running.

Options:

/H Save everything in memory to your hard disk, and shutdown to save power. The desktop is restored to its original state when the computer is restarted.

/K Lock the workstation. To unlock, the user must log in.

/L Log off Windows, but do not reboot. This option is equivalent to selecting Shutdown from the Start menu, then selecting "Close all programs and log on as a different user" in the shutdown dialog.

/M Switch the display to low power (M0) or shut off the display (M1 -- will not work on all systems). This option will not reboot the computer unless you also include /R.

/P Log off Windows and turn off the computer.

/R Reboots the system. This is the default, but is required if you specify /M0 or /M1 and also want to reboot.

/S Shut down the system, but do not reboot. This is equivalent to selecting Shutdown from the Start menu, then selecting "Shut down the computer" in the shutdown dialog.

/V Prompt for confirmation (**Y** or **N**) before acting.

/W Save power by turning off the monitor and hard disks. When the computer comes out of standby, the desktop is restored to its original state.

7.92 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) 336 /P(rompt) 336
/E (no error messages) 336 /Q(uiet) 336

Usage:

If you don't specify any drives (or paths), RECYCLE will delete (or display) everything in the recycle bin for all local drives.

Options:

/D Empty the recycle bin for the specified drive(s).

/E Suppress all non-fatal error messages, such as "File Not Found." Fatal error messages, such as "Drive not ready," will still be displayed. This option is most useful in batch files.

/P Prompt the user to confirm each delete operation.

/Q Don't display the name of the recycle bin(s). This option is most often used in batch files.

7.93 REM

Purpose: Put a comment in a batch file.

Format: REM [comment]

comment The text to include in the batch file.

Usage:

The REM command lets you place a remark or comment in a batch file. Batch file comments are useful for documenting the purpose of a batch file and the procedures you have used. For example:

```
rem This batch file provides a
rem menu-based system for accessing
rem word processing utilities.
rem
rem Clear the screen and get selection
cls
```

REM must be followed by a space or tab character, then the comment. Comments can be up to 8,191 characters long. The command processor will ignore everything on the line following the REM, including quotes, redirection symbols, and other commands (see below for the exception to this rule).

If ECHO is ON, the comment is displayed. Otherwise, it is ignored. If ECHO is ON and you don't want to display the line, preface the REM command with an at sign [@].

You can also place a comment in a batch file by starting the comment line with two colons [**::**]. In essence this creates a batch file "label" without a valid label name.

You can use REM to create a zero-byte file if you use a redirection symbol immediately after the REM command. For example, to create the zero-byte file *C:\xyz*:

```
rem>xyz
```

(This capability is included for compatibility with and CMD.EXE. A simpler method for creating a zero-byte file with the command processor is to use **>filename** as a command, with no actual command before the [**>**] redirection character.)

7.94 REN / RENAME

Purpose: Rename files or subdirectories.

Format: REN [/A:[[-][+]rhsadecijopt] /E /I"text" /N[et] /P /Q /S /T] [@*file*] old_name... new_name
or
RENAME [/A:[[-][+]rhsadecijopt] /E /I"text" /N[et] /P /Q /S /T] [@*file*] old_name... new_name

old_name	Original name of the file(s) or subdirectory.
new_name	New name to use, or new path on the same drive.
@file	A text file containing the names of the source files to rename, one per line (see @file lists 32 for details).

/A: (Attribute select) 339
/E (No error messages) 339
/I"text" (match description) 339
/N (Disable) 339
/P(rompt) 339
/Q(uiet) 339
/S(ubdirectory) 339
/T(otal) 340

See also: COPY 216 and MOVE 308.

File Selection:

Supports attribute switches 28, extended wildcards 19, ranges 22, multiple file names 29, delayed variable expansion 31, and include lists 30. Use wildcards with caution on LFN volumes; see LFN File Searches 31 for details.

Internet: Can be used with FTP/HTTP Servers 42 and HTTP/HTTPS servers.

Usage:

REN and RENAME are synonyms. You may use either one.

REN lets you change the name of a file or a subdirectory, or move one or more files to a new subdirectory on the same drive. (If you want to move files to a different drive, use MOVE.)

In its simplest form, you give REN the ***old_name*** of an existing file or subdirectory and then a ***new_name***. The ***new_name*** must not already exist — you can't give two files the same name (unless they are in different directories). The first example renames the file *MEMO.TXT* to *MEM.TXT*. The second example changes the name of the *\WORD* directory to *\WP*:

```
rename memo.txt mem.txt
rename /s \word \wp
```

When you rename files or directories on an LFN drive, you must quote any names which contain white space or special characters.

You can also use REN to rename a group of files that you specify with wildcards, as multiple files, or in an include list. When you do, the ***new_name*** must use one or more wildcards to show what part of each filename to change. Both of the next two examples change the extensions of multiple files to *.SAV*:

```
ren config.nt autoexec.nt tcstart.btm *.sav
ren *.txt *.sav
```

REN can move files to a different subdirectory on the same drive. When it is used for this purpose, REN requires one or more filenames for the ***old_name*** and a directory name for the ***new_name***:

```
ren memo.txt \wp\memos\
ren oct.dat nov.dat \data\save\
```

The final backslash in the last two examples is optional. If you use it, you force REN to recognize the last parameter as the name of a directory, not a file. The advantage of this approach is that if you accidentally mistype the directory name, REN will report an error instead of renaming your files in a way that you didn't intend.

REN can also move files to a new directory and change their name at the same time if you specify both a path and file name for *new_name*. In this example, the files are renamed with an extension of *.SAV* as they are moved to a new directory:

```
ren *.dat \data\save\*.sav
```

If you use REN to rename a directory, the ***new_name*** must normally be specified explicitly, and cannot contain wildcards. You can override this restriction with ***/S***. When you rename a directory the extended directory search 14 database will be automatically updated to reflect the change.

You can also rename a subdirectory to a new location in the directory tree on the same physical drive (sometimes called "prune and graft"). You must specify the new name explicitly, not just give the path. For example, if the ***D:\4NT*** directory contains a subdirectory *TEST*, you can rename TEST to be a subdirectory of the root directory like this:

```
[d:\4NT] ren TEST \TEST\
```

REN does not change a file's attributes, 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 42 and IFTP 288.

Note: The wildcard expansion process will attempt to allow both CMD.EXE-style "extension" matching (assumes only one extension, at the end of the word) and the advanced ***4NT***/***TC*** 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 28 for information on the attributes which can follow **/A:**. Do not use /A: with @file lists 32. See @file lists 32 for details.

/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 19 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 32 for details.

/N Do everything except actually rename the file(s). **/N** displays how many files would be renamed. This option is useful for testing what a REN command will actually do.

A **/N** with one of the following arguments has an alternate meaning:

- **e** Don't display errors.
- **t** Don't update the CD / CDD extended directory search 14 database *(JPSTREE.IDX)*.

/P Prompt the user to confirm each rename operation. Your options at the prompt are explained in detail under Page and File Prompts 41.

/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 279 REN.

/T Don't display filenames as they are renamed, but report the number of files renamed. See also **/Q**.

7.95 RETURN

Purpose: Return from a GOSUB (subroutine) in a batch file.

Format: RETURN [value]

value The numeric exit code to return to the command processor.

See also: GOSUB 280.

Usage:

The command processor allows subroutines in batch files.

A subroutine begins with a label (a colon followed by one or more words) and ends with a RETURN command.

The subroutine is invoked with a GOSUB command from another part of the batch file. When a RETURN command is encountered the subroutine terminates, and execution of the batch file continues on the line following the original GOSUB. If RETURN is encountered without a GOSUB, the command processor will display a "Missing GOSUB" error message.

You cannot execute a RETURN from inside a DO 246 loop.

The following batch file fragment calls a subroutine which displays the files in the current directory:

```
echo Calling a subroutine
gosub subr1
echo Returned from the subroutine
quit
:subr1
dir /a/w
return
```

If you specify a ***value***, RETURN will set the internal exit code to that value. That exit code should be tested immediately upon return from the subroutine and before it is reset by another command. For information on exit codes from internal commands, see the _? 410 variable.

7.96 REXEC

Purpose: Remotely execute commands

Format: REXEC [/H host /U name /P password /Rn /Tn] host [/L userid] command ...

command The command to execute

/H(host name) 341
/L (user ID) 341
/P(assword) 341
/R(emote port) 341
/T (firewall type) 341
/U(sername) 341

Usage:

REXEC allows remote execution of commands on any system with the rexec service installed. Press ^C to disconnect from the other system.

If you don't specify a username, RSHELL will use the current username. You can provide a password on the command line by appending it to the username (i.e., "User:Password"). If you don't provide a password, REXEC will prompt for it.

If you want to do redirection on the remote system, enclose the argument list in double quotes. For example:

```
REXEC /H host /U user /P password "command | command2"
```

The double quotes will be removed before passing the commands to the remote system.

Note: Windows does not include the rshell service, so you will need to get one from a third-party and install it on the remote system before executing RSHELL.

Options:

/H Firewall host name

/L User name (ID)

/P Firewall user password

/R Remote port number

/T Firewall type, where ***n*** is:

0 - No firewall (default setting)
1 - Connect through a tunneling proxy
2 - Connect through a SOCKS4 Proxy
3 - Connect through a SOCKS5 Proxy

/U Firewall user name

7.97 RSHELL

Purpose: Remotely execute commands

Format: RSHELL [/H host /U name /P password /Rn /Tn] host [/L userid] command ...

command The command to execute

/H(ost name) 342 /R(emote port) 342
/L (user ID) 342 /T (firewall type) 342
/P(assword) 342 /U(sername) 342

Usage:

RSHELL allows remote execution of commands on any system with the rshell service installed. Press ^C to disconnect from the other system.

If you don't specify a username, RSHELL will use the current username.

If you want to do redirection on the remote system, enclose the argument list in double quotes. For example:

```
RSHELL /H host /U user /P password "command | command2"
```

The double quotes will be removed before passing the commands to the remote system.

Note: Windows does not include the rshell service, so you will need to get one from a third-party and install it on the remote system before executing RSHELL.

Options:

/H Firewall host name

/L User name

/P Firewall user password

/R Remote port number

/T Firewall type, where ***n*** is:

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

7.98 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 253, ECHOS and ECHOSERR 255, SCRPUT 344, TEXT 376, and VSCRPUT 391.

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 344 or VSCRPUT 391. SCREEN always leaves the cursor at the end of the displayed text.

The ***row*** and ***column*** values are zero-based, so on a 25 line by 80 column display, valid ***rows*** are 0 - 24 and valid ***columns*** are 0 - 79. SCREEN checks for a valid ***row*** and ***column***, and displays a "Usage" error message if either value is out of range.

(TC) In ***Take Command***, the maximum ***row*** value is determined by the current height of the ***TC*** window, and the maximum ***column*** value is determined by the current virtual screen width (see Resizing the Take Command Window 84 for more information).

You can also specify the ***row*** and ***column*** as offsets from the current cursor position. Begin the value with a plus sign [**+**] to move the cursor down or to the right, or with a minus sign [**-**] to move the cursor up or to the left. This example prints a string 3 lines above the current position, in absolute column 10:

```
screen -3 10 Hello, World!
```

you specify 999 for the ***row***, SCREEN will center the text vertically on the display. If you specify 999 for the ***column***, SCREEN will center the text horizontally. This example prints a message at the center of the command processor window:

```
screen 999 999 Hello, World
```

7.99 SCRIPT

Purpose: Run a script using an Active Scripting engine.

Format: SCRIPT [/E engine] [filename ...]

/E(ngine) 344

engine The name of the scripting engine

Usage:

If you don't specify any arguments, SCRIPT will display the installed engines.

See also the @SCRIPT 488 variable function.

4NT and ***TC*** have a COM interface to allow the script to call back into the command processor. The methods are:

shell.exec("command") - execute the specified command (internal or external)

shell.write("string") - write the string to stdout

shell.writeLn("string") - write the string to stdout with a CR/LF

shell.alert("text") - pop up a message box

Options:

/E If the script doesn't have a recognized extension (i.e., **.vbs**, **.pls**, etc.) you will need to specify the engine SCRIPT should use to execute the script.

7.100 SCRPUT

Purpose: Position text on the screen and display it in color.

Format: SCRPUT *row col* [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

See also: ECHO and ECHOERR 253, ECHOS and ECHOSERR 255, SCREEN 342, TEXT 376, and VSCRPUT 391.

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 552 for details.

The ***row*** and ***column*** values are zero-based, so on a 25 line by 80 column display, valid ***rows*** are 0 - 24 and valid ***columns*** are 0 - 79. ***(TC)*** The maximum ***row*** value is determined by the current height of the ***TC*** window. The maximum ***column*** value is determined by the current virtual screen width (see Resizing the Take Command Window 84 for more information).

(TC) SCRPUT checks for a valid ***row*** and ***column***, and displays a "Usage" error message if either value is out of range.

You can also specify the ***row*** and ***column*** as offsets from the current cursor position. Begin the value with a plus sign [**+**] to move down the specified number of rows or to the right the specified number of columns, or with a minus sign [**-**] to move up or to the left.

If you specify 999 for the ***row***, SCRPUT will center the text vertically in the command processor window. If you specify 999 for the ***column***, SCRPUT will center the text horizontally.

SCRPUT does not move the cursor when it displays the ***text***.

The following batch file fragment displays part of a menu, in color:

```
cls white on blue
scrput 3 10 bri whi on blu Select an option:
scrput 6 20 bri red on blu 1 - Word Processing
scrput 7 20 bri yel on blu 2 - Spreadsheet
scrput 8 20 bri gre on blu 3 - Communications
scrput 9 20 bri mag on blu 4 - Quit
```

7.101 SELECT

Purpose: Interactively select files for a command.

Format: SELECT [/1 /A[[:][-][+]rhsadecijopt] /C /D /E /H /I"text" /J /L /O[:][-]acdeginorsu /T:acw /X /Z] [command] ... (files...)...

command 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 347
/A(ttribute select) 347
/C(ompression) 348
/D(isable color coding) 348
/E (use upper case) 348
/H(ide dots) 348
/I"text" (match descriptions) 348
/J(ustify names) 348
/L(ower case) 348
/O(rder) 348
/T(ime) 348
/X (display short names) 348
/Z (FAT format) 348

File Selection

Supports extended wildcards 19, ranges 22, multiple file names 29, and include lists 30. Ranges **must** appear immediately after the SELECT keyword.

Internet: Can be used with FTP servers. See Using FTP/HTTP Servers 42.

Usage:

SELECT allows you to select files for internal and external commands by using a "point and shoot" display. You can have SELECT execute a command once for each file you select, or have it create a list of files for a command to work with. The ***command*** can be an internal command, an alias, an external command, or a batch file.

If you use parentheses around the ***files***, SELECT executes the ***command*** once for each file you have selected. During each execution, one of the selected files is passed to the ***command*** as a parameter. If you use square brackets around *files*, the SELECTed files are combined into a single list, separated by spaces. The command is then executed once with the entire list presented as part of its command line parameters.

SELECT can also select files on FTP servers. For example:

```
select del (ftp://ftp.domain.com/)
```

You can also use the IFTP command to start an FTP session on a server, and then use an abbreviated syntax to specify the files and directories you want. For more information, see Using FTP/HTTP Servers 42 and IFTP 288.

Using the SELECT File List

When you execute the SELECT command, the file list is displayed in a full-window format which includes a top-line status bar and shows the command to be executed, the number of files marked, and the number of Kbytes in those files.

SELECT supports the mouse for selecting and scrolling the list. You can also use the cursor up, cursor down, PgUp, and PgDn keys to scroll through the file list. You can also use character matching

to find specific files, just as you can in any popup window 536. While the file list is displayed you can enter any of the following keys to select or unselect files, display files, execute the command, or exit:

space	Select a file, or unselect a marked file.
+	Select a file (all products), or unselect a marked file ***(4NT)***.
-	Unselect a marked file.
*****	Reverse all of the current marks (except those on subdirectories). If no files have been marked you can use ***** to mark all of the files.
/	Unselect all files.
Ctrl-L	***(4NT)*** View the current highlighted file with LIST 298. When you exit from LIST, the SELECT screen will be restored.
Enter	Execute the command with the marked files, or with the currently highlighted file if no files have been marked.
Esc	Skip the files in the current display and go on to the next file specification inside the parentheses or brackets (if any).
Ctrl-C or **Ctrl-Break**	Cancel the current SELECT command entirely.

On FAT drives the file list is shown in standard FAT directory format, with names at the left and descriptions at the right. On LFN drives the format is similar but more space is allowed for the name, and the description is not shown. In this format long names are truncated if they do not fit in the allowable space. For a short-name format (including descriptions) on long filename drives, use the ***/X*** and / or ***/Z*** switches.

When displaying descriptions in the short filename format, SELECT adds a right arrow at the end of the line if the description is too long to fit on the screen. This symbol will alert you to the existence of additional description text. You can use the left and right arrow keys to scroll the description area of the screen horizontally and view the additional text.

(4NT) You can set the default colors used by SELECT on the Colors tab 154 of the configuration dialogs, or with the SelectColors 143 and SelectStatBarColors 143 directives in the .INI file 91. If SelectColors is not used, the SELECT display will use the current default colors. If SelectStatBarColors is not used, the status bar will use the reverse of the SELECT colors.

Creating SELECT Commands

In the simplest form of SELECT, you merely specify the command and then the list of files from which you will make your selection(s). For example:

```
select copy (*.com *.exe) a:\
```

will let you select from among the *.COM* files on the current drive, and will then invoke the COPY command to copy each file you select to drive A:. After the *.COM* files are done, the operations will be repeated for the *.EXE* files.

If you want to select from a list of all the *.COM* and *.EXE* files mixed together, create an include list 30 inside the parentheses by inserting a semicolon:

```
select copy (*.com;*.exe) a:\
```

Finally, if you want the SELECT command to send a single list of files to COPY, instead of invoking COPY once for each file you select, put the file names in square brackets instead of parentheses:

```
select copy [*.com;*.exe] a:\
```

If you use brackets, you have to be sure that the resulting command (the word COPY, the list of files,

and the destination drive in this example) does not exceed the command line length limit 71. 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 22. The range(s) must be placed immediately after the word SELECT. If the ***command*** is an internal command that supports ranges, an independent range can also be used in the ***command*** itself.

You cannot use command grouping to make SELECT execute several commands, because SELECT will assume that the parentheses are marking the list of files from which to select, and will display an error message or give incorrect results if you try to use parentheses for command grouping instead. (You can use a SELECT command inside command grouping parentheses, you just can't use command grouping to specify a group of commands for SELECT to execute.)

Advanced Topics

If you don't specify a command, the selected filename(s) will become the command. For example, this command defines an alias called UTILS that selects from the executable files in the directory **C:\UTIL**, and then executes them in the order marked:

```
alias utils select (c:\util\*.com;*.exe;*.btm;*.bat)
```

If you want to use filename completion 58 to enter the filenames inside the parentheses, type a space after the opening parenthesis. Otherwise the command line editor will treat the open parenthesis as the first character of the filename.

With the **/I** option, you can select files based on their descriptions. SELECT will display files if their description matches the text after the **/I** switch. The search is not case sensitive. You can use wildcards and extended wild cards as part of the text.

When sorting file names and extensions for the SELECT display, the command processor normally assumes that sequences of digits should be sorted numerically (for example, the file DRAW2 would come before DRAW03 because 2 is numerically smaller than 03), rather than strictly alphabetically (where DRAW2 would come second because "2" comes after "0"). You can defeat this behavior and force a strict alphabetic sort with the **/O:a** option.

Options:

/1 Only allow one selection.

/A[:] Select only those files that have the specified attribute(s) set. See Attribute Switches 28 for information on the attributes which can follow **/A:**.

/C Display per-file and total compression ratios on compressed drives. The compression ratio is displayed instead of the file description. The ratio is left blank for directories and files with a length of 0 bytes, and for files on non-compressed drives. The compression ratios will not be visible on LFN drives unless you use **/Z** to switch to the short filename format. Only compressed NTFS drives are supported. See DIR /C 233 for more details on how compression ratios are calculated.

/D ***(4NT)*** Temporarily turn off directory color coding.

/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 19 and extended wildcards. The search text must be enclosed in double quotes, and must follow the **/I** immediately, with no intervening spaces. You can select all filenames that have a description with **/I"[?]*"**, or all filenames that do not have a description with **/I"[]"**.

/J Justify (align) filename extensions and display them in the FAT format.

/L Display file and directory names in lower case.

/O Set the sort order for the files. The order can be any combination of the following options:

- **n** Sort by filename (this is the default)
- \- Reverse the sort order for the next option.
- **a** Sort names and extensions in standard ASCII order, rather than sorting numerically when digits are included in the name or extension.
- **c** Sort by compression ratio (the least compressed file in the list will be displayed first). For information on supported compression systems see **/C** above.
- **d** Sort by date and time (oldest first).
- **e** Sort by extension.
- **g** Group subdirectories together.
- **i** Sort by the file description (ignored if **/C** or **/O:c** is also used).
- **o** Sort by owner
- **r** Reverse the sort order for all options.
- **s** Sort by size.
- **u** Unsorted.

/T:acw Specify which of the date and time fields on an LFN drive should be displayed and used for sorting:

- **a** Last access date and time (access time is not saved on VFAT and FAT32 volumes).
- **c** Creation date and time.
- **w** Last write date and time (default).

/X Display short filenames in FAT format (like **/Z**), on LFN drives.

/Z Display a directory on an LFN drive in the old-style format, with the filename at the left and the description at the right. Long names will be truncated to 12 characters; if the name is longer than 12 characters, it will be followed by a right arrow.

7.102 SENDMAIL

Purpose: Send an email message.

Format: SENDMAIL [/A file1 [/A file2 ...] /D /Eaddress /H"header: value" /In /M /Pn /R /Sn /V] "address[,address...] [cc:address[,address] bcc:address[,address...]]" subject [text | @msgfile]

file1...	The attachment files.
address 349	The destination email address.
subject 349	The subject line.
text 349	The message to send.
msgfile 349	The file containing the message body.

/A 350 file	Attachment	/M 350	CRAM-MD5 authentication
/D 350	Delivery Confirmation	/P 350 n	Priority
/E 350	Reply-to address	/R 350	Send read receipt
/H 350	Send custom header	/Sn 350	Sensitivity
/I 350 n	Importance	/V 349	Verbose

See also: SNPP 363 and SMPP 362.

Usage:

SENDMAIL sends an email message from ***TC*** or ***4NT*** via SMTP. The text of the message can be entered either on the command line or read from a text file.

Before you can use SENDMAIL, you must either set the MailServer 132 and MailUser 132 directives, or have a default account in the registry. Depending on your system configuration, you may also need to start an Internet connection before you use SENDMAIL.

A SENDMAIL message has three required parts: an address 349, a subject 349, and message 349. Optionally it may also have attachments 350.

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:
```

SENDMAIL uses the current values of the directives MailAddress 132, MailPassword 132, MailPort 132, MailServer 132 and MailUser 132. If you wish to temporarily alter those values, use the OPTION 317 command :

```
OPTION //MailAddress="Bill Shakespeare <bard@globe.com>"
SENDMAIL ...
```

Note: SENDMAIL support is provided by **ipworks6.dll** and **ipwssl6.dll** included in the ***4NT*** and ***TC*** distribution packages). These files are installed by default in the command processor's directory.

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.

/In Set the Importance where ***n*** is:

- **1** High
- **2** Normal (default)
- **3** Low

/M Use CRAM-MD5 authentication.

/Pn Set the Priority where ***n*** is:

- **0** Unspecified (default)
- **1** Normal
- **2** Urgent
- **3** Non Urgent

/R (Read receipt) : Send a read receipt.

/Sn Set the message sensitivity. The values are:

- **1** Personal
- **2** Private
- **3** CompanyConfidential

/V Show all the interaction with the server, except the message header and message body

text.

7.103 SET

Purpose: Display, create, modify, or delete environment variables.

Format: Display mode:
SET [/D /E /P /S /U /V /X] [wildname]

Definition mode:
SET [/A /D /S /U /V /E /R [file...] | name=value | prompt]

Deletion mode:
SET [/D /S /U /V /E] name

file	One or more input files to read variable definitions from.
name	The name of the environment variable
value	The new value for the variable, separated from name by space[s]
prompt	Optional input prompt for the ***/P name=*** option
wildname	Name of variable[s] to be displayed. May contain ******* wildcard unless displaying registry variables

/A 353	Arithmetic	/S 354	System
/D 353	Default	/U 354	/User
/E 353	Environment, too	/V 354	Volatile
/P 353	Pause or Prompt	/X 354	Override VariableExclude
/R 354	Read from file(s)		

See also: ESET 258, UNSET 388, and SettingChange 143.

Usage:

Every program and command inherits an environment 395, 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. The command processor itself uses several environment variables 395.

(TC) You can also create or modify environment variables with the Environment Window 81. All of the information in this section also applies to variables defined via the dialog, unless otherwise noted.

If you simply type the SET command with no options or 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 353, /S 354, /U 354, and /V 354).

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 command processor internal variables 402, 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 258 command. To remove variables from the environment, use UNSET 388, or type SET, followed by the variable name and an equal sign:

```
set mine=
```

The variable's ***name*** is limited to a maximum of 80 characters. ***Name*** and ***value*** together cannot be longer than 8,191 characters.

Note: You cannot use SET to modify GOSUB variables 280.

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 353, /S 354, /U 354 and /V 354 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 353, /S 354, /U 354 or /V 354 **will not set** the variable in the local environment unless you also use the /E 353 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 SettingChange 143 .INI directive is set, the command processor will monitor the WM_SETTINGCHANGE message and update the environment from the User, Volatile, and System registry entries. The update is done whenever the command processor displays the prompt (to prevent the environment from changing in the middle of a command).

Options:

/A Evaluate the arithmetic expression on the right of the equal sign, place the result in the environment, and display it. For example, this command adds 2 and 2, and places the result in the environment variable **VAR**:

```
set /a var=2+2
```

/A interprets non numeric strings in ***value*** as environment variable names whether or not preceded by a percent sign **%**, and replaces them with their respective ***values***. For example, this sequence will set **Y** to **4**:

```
set x=2
set /a y=x+2
```

You can use @EVAL 451 to perform the same task; SET /A is included for compatibility with **CMD.EXE**. Unlike @EVAL 451, use of the >> or << shift operators in SET /A requires disabling their interpretation as redirection symbols by using SETDOS 354 /X-6.

/D Create/modify/delete a **default** variable in the registry (HKU\.DEFAULT\Environment).

/E When used together with one of /D 353, /S 354, /U 354, or /V 354, set both the registry variable and the local environment variable.

/P When used without a variable name, wait for a key to be pressed after each screen page before continuing the display. Your options at the prompt are explained in detail under Page and File Prompts 41.

When used with a variable name and an optional prompt string, e.g. set /p ***myvar***=Enter

value, emulates the **CMD.EXE** behavior of allowing entry of a value for the variable. This is provided for compatibility reasons only. For more flexibility, use the ESET 258 or INPUT 293 command.

/R Read environment variables from a file. This is much faster than loading variables from a batch file with multiple SET commands. Each entry in the file must fit within the command line length limit 71 for the command processor. The file is in the same format as the SET display (*i.e.*, ***name=value***), so SET /R can accept as input a file generated by redirecting SET output. For example, the following commands will save the environment variables to a file, and then reload them from that file:

```
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 69. However, you cannot use this method to exceed the command line length limit. You can also add comment lines to the file by starting each with a colon **:**. You can also use other special characters, e.g., redirection and pipe symbols (**<>|**), without the need for special handling (e.g., escaping). If you reference the value of another variable in ***value*** (e.g., **x=%path;c:\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 571.

/S Create/modify/delete a **system** variable in the registry (`HKLM\System\CurrentControlSet\Control\Session Manager\Environment`).

/U Create/modify/delete a **user** variable in the registry (`HKCU\Environment`).

/V Create/modify/delete a **volatile** variable in the registry (`HKCU\Volatile Environment`).

/X Override the **VariableExclude** variable and display all matching variables.

7.104 SETDOS

Purpose: Display or set the command processor configuration.

Format: SETDOS [/A? /C? /D? /E? /Fn.n /G?? /I[+|-] *command* /M? /N? /P? /S?:? /V? /X[+|-]n]

/A(NSI) 355
/C(ompound) 355
/D(escriptions) 355
/E(scape character) 355
/F(ormat for @EVAL) 356
/G (numeric separators) 356
/I(nternal) 356
/M(ode for editing) 356
/N(o clobber) 356
/P(arameter character) 356
/S(hape of cursor) 356
/V(erbose) 357
/X (expansion, special characters) 357

See also: OPTION 317.

Usage:

SETDOS allows you to customize certain aspects of the command processor to suit your personal tastes or the configuration of your system.

You can display the value of all SETDOS options by entering the SETDOS command with no parameters.

Most of the SETDOS options can be initialized when the command processor executes the configuration directives 99 in the .INI file 91, and can also be changed from the configuration dialogs 151. The name of the corresponding directive is listed with each option below; if none is listed, that option cannot be set from the *.INI* file. You can also define the SETDOS options in your **TCSTART *(TC)*** or **4START *(4NT)*** or other startup file (see Automatic Batch Files 8), in aliases, or at the command line.

Note: The functionality of the "/Y" option ("debug", no longer supported) of previous versions has been moved to the BDEBUGGER 201 command.

Inheritance

When a new instance of the command is started, it inherits the SETDOS characteristics set by the most recently started instance of the command processor.

Options:

/A [ANSI 105] This option determines whether ANSI X3.64 support 40 is enabled. **/A1** enables ANSI X3.64 string processing. The default of **/A0** disables ANSI X3.64 strings. See the ANSI X3.64 Commands Reference 550 for a list of the ANSI X3.64 sequences supported by ***4NT*** and ***TC***. See also: the ANSI 105 directive and the _ANSI 411 internal variable.

/C [CommandSep 110] 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 36 characters (**| > <**), or a space, tab, comma, or equal sign as the command separator. The command separator is saved by SETLOCAL 358 and restored by ENDLOCAL 256. The following example changes the separator character to a tilde [**~**]:

```
setdos /c~
```

/D [Descriptions 115 and DescriptionName 115] This option controls whether file processing commands like COPY 216, DEL 225, MOVE 308, and REN 337 process file descriptions along with the files they belong to. **/D1** turns description processing on, which is the default. **/D0** turns description processing off. See also: the Descriptions 115 directive.

You can also use **/D** to set the name of the hidden file in each directory that contains file descriptions. To do so, follow **/D** with the filename in quotes:

```
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 [EscapeChar 117] This option sets the character used to suppress the normal meaning of the following character. Any character following the escape character 69 will be passed unmodified to the command. The default escape character is a caret [**^**]. You cannot use

any of the redirection 36 characters (**| > <**) or a space, tab, comma, or equal sign as the escape character. The escape character is saved by SETLOCAL 358 and restored by ENDLOCAL 256. Certain characters (**b**, **c**, **e**, **f**, **k**, **n**, **q**, **r**, **s**, and **t**) have special meanings when immediately preceded by the escape character.

/F [EvalMax 117, EvalMin 117] This option lets you set default decimal display precision for the @EVAL 451 variable function. The maximum precision is 20 digits to the left of the decimal point and up to 10 digits to the right of the decimal point.

The 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 EvalMax 117 and EvalMin 117 directives to set the precision when the command processor starts; see the @EVAL 451 function if you want to set the display precision for a single computation.

/G [DecimalChar 113, ThousandsChar 147] This option sets the decimal 113 and thousands 147 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 451, numeric IF 286 and IFF 287 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 the command processor to use an external command:

```
setdos /i-list
```

To re-enable all disabled commands use **/I***.

/M [EditMode 116] This option controls the initial line editing mode. To start in overstrike mode at the beginning of each command line, use **/M0** (the default in ***4NT***). To start in insert mode, use **/M1** (the default in ***TC***). See also: the EditMode 116 directive.

/N [NoClobber 133] This option controls output redirection 36. **/N0** means existing files will be overwritten by output redirection (with **>**) and that appending (with **>>**) does not require the file to exist already. This is the default. **/N1** means existing files may not be overwritten by output redirection, and that when appending the output file must exist. A **/N1** setting can be overridden with the [**!**] character. See also: the NoClobber 133 directive.

/P [ParameterChar 135] This option sets the character used after a percent sign to specify all or all remaining command line parameters in a batch file 162 or alias 187. The default value is the dollar sign [**$**]. The parameter character is saved by SETLOCAL 358 and restored by ENDLOCAL 256.

/S ***(4NT)*** [CursorOver 112, CursorIns 112] The size is entered as a percentage of the total character height. The default values are 10:100 (a 10% underscore cursor for overstrike

mode, and a 100% block cursor for insert mode). Because of the way video drivers remap the cursor shape, you may not get a smooth progression in the cursor size from 1% - 100%. (You can disable the cursor by specifying a size of 0:0.)

(TC) [CursorOver 112, CursorIns 112] The size is entered as a percentage of the total character width. The default values are 100:15 (a 100% or block cursor for overstrike mode, and a 15% or thin line cursor for insert mode).

If either value is -1, the command processor will not attempt to modify the cursor shape at all. You can retrieve the current cursor shape values with the **%_CI** and **%_CO** internal variables. See also the CursorOver 112 and CursorIns 112 directives.

/V [BatchEcho 107] This option controls the default for command echoing in batch files.

/V0 disables echoing of batch file commands unless ECHO 253 is explicitly set ON.

/V1, the default setting, enables echoing of batch file commands unless ECHO 253 is explicitly set OFF. See also: the BatchEcho 107 directive

/X[+|-]n (expansion and special characters) This option enables and disables alias and environment variable expansion, and controls whether special characters have their usual meaning or are treated as text. It is most often used in batch files to process text strings which may contain special characters.

The features enabled or disabled by **/X** are numbered. All features are enabled when the command processor starts, and you can re-enable all features at any time by using **/X0**. To disable a particular feature, use **/X-n**, where **n** is the feature number from the list below. To re-enable the feature, use **/X+n**. To enable or disable multiple individual features, list their numbers in sequence after the **+** or **-** (*e.g.* **/X- 345** to disable features 3, 4, and 5).

The features are:

1 All alias expansion
2 *Nested* alias expansion only
3 All variable expansion (includes environment variables, batch file parameters, variable function evaluation, and alias parameters)
4 *Nested* variable expansion only
5 Multiple commands, conditional commands, and piping (affects the command separator, **||**, **&&**, **|**, and **|&**)
6 Redirection (affects **<** , **>**, **>&**, **>&>**, etc.)
7 Quoting (affects back-quotes [`] and double quotes [**"**]) and square brackets)
8 Escape character
9 Include lists 30

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 358 command will save the current SETDOS /X values for ENDLOCAL 256 to restore.

7.105 SETLOCAL

Purpose: Save a copy of the current disk drive, directory, environment, alias list, and special characters.

Format: SETLOCAL

See also: ENDLOCAL 256.

Usage:

SETLOCAL is valid only in batch files. It will save :

- the default disk drive and directory
- the environment,
- the alias list
- the special character set (command separator, escape character, parameter character, decimal separator, and thousands separator)
- the SETDOS /X 354 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 256 command, or just by exiting the batch file.

SETLOCAL does not save the command history or user functions.

For example, this batch file fragment saves everything, removes all aliases so that user aliases will not affect batch file commands, changes the disk and directory, changes the command separator, runs a program, and then restores the original values:

```
setlocal
unalias *
cdd d:\test
setdos /c~
program ~ echo Done!
endlocal
```

SETLOCAL and ENDLOCAL may be nested up to 16 levels deep in each batch file. You can also have multiple SETLOCAL / ENDLOCAL pairs within a batch file, and nested batch files can each have their own SETLOCAL / ENDLOCAL pairs.

SETLOCAL does not override the LocalAliases 130=No setting. Consequently changing aliases inside a SETLOCAL/ENDLOCAL pair affects the definition of aliases of other concurrently executing sessions of the current command processor.

You cannot use SETLOCAL in an alias or at the command line.

An ENDLOCAL is performed automatically at the end of a batch file. If you invoke one batch file from another without using CALL 209, 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 256 for details.

7.106 SHIFT

Purpose: Allows the use of more than 512 parameters in a batch file, or iterating through its parameters. This command can be used only in batch files.

Format: SHIFT [[-]n | /n]

n Number of positions to shift (an unsigned number), or the position of the parameter to be deleted.

Usage:

SHIFT is provided for compatibility with batch files written for CMD.EXE, where it was used to access more than 10 parameters. ***4NT*** and ***TC*** support 512 parameters (%0 to %511), so you may not need to use SHIFT for batch files running exclusively under ***4NT*** or ***TC***.

SHIFT *n* moves each of the batch file parameters ***n*** positions to the left. The default value for ***n*** is **1**. For example, SHIFT (with no parameters) makes the parameter **%1** become to **%0**, the parameter **%2** becomes **%1**, etc.

SHIFT *-n* moves parameters to the right, but it is limited to moving them back to their position on entry to the batch file.

This form of SHIFT also affects the special parameters **%n$**, **%$** and **%#** (number of command parameters). However, for compatibility with **CMD.EXE**, this form of the SHIFT command does not alter the contents or order of the parameters returned by **%*** . See Batch File Parameters 165 for details.

For example, create a batch file called TEST.BAT:

```
echo %1 %2 %3 %4
shift
echo %1 %2 %3 %4
shift 2
echo %1 %2 %3 %4
shift -1
echo %1 %2 %3 %4
```

Executing the command below produces the following results:

```
[c:\] test one two three four five six seven
one two three four
two three four five
four five six seven
three four five six
```

SHIFT /***n*** This form of the command irreversibly deletes parameter ***%n*** from the command tail, and shifts all parameters originally to its right **1** position to the left. For example,

```
shift /2
```

leaves parameters **%0** and **%1** unchanged, and moves the value of **%3** to position **%2**, **%4** to **%3**, etc.

This form of SHIFT also affects the special parameters **%n$**, **%$** and **%#** (number of batch file parameters), and unlike the first form, it also affects **%***. See Batch File Parameters 165 for details.

7.107 SHORTCUT

Purpose: Create or display a shortcut.

Format: Creation mode 360
SHORTCUT command args dir desc link mode [iconfile [iconoffset [hotkey]]]

Display mode 361
SHORTCUT link

command	Command the shortcut executes
args	Command line parameters for command
dir	Starting directory
desc	Description
link	Filename of the .LNK or .PIF file.
mode	Initial window mode: 1=normal, 2=minimized, 3=maximized
iconfile	File containing the icon to use
iconoffset	Icon offset within iconfile
hotkey	Hotkey to invoke the shortcut

Usage:

Creation Mode

SHORTCUT creates a Windows shortcut file and places it in the specified directory. You can run any Windows shortcut from the command processor by entering the name of the *.LNK* or *.PIF* file on the command line.

SHORTCUT requires a minimum of 6 parameters. To leave a parameter blank, enter an empty string (2 double quotes **""** in its place. Any parameter must be enclosed in double quotes if it includes white space or other special characters.

Command is the full path of the executable file to start, or the data file or folder to open. If it is a data file, its extension must be associated with an executable command (see ASSOC 197) 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 the command processor exits immediately when the batch file is completed.

The ***dir*** parameter is the path of the directory to which you want Windows to switch when the command starts. If you don't care which directory is used, you can omit this parameter by entering **""** in its place.

Desc provides a description that is stored internally in the shortcut. It is displayed when the cursor is moved to the shortcut. If you omit the description, enter **""** in its place.

The ***link*** parameter is the drive, path, name and extension of the shortcut file you want to create. The drive and path portion is interpreted according to the usual rules - missing elements default to the

current defaults, path is relative to the current default unless it starts with \. The file extension must be **.LNK**, unless you are creating a shortcut to a DOS command, in which case the extension must be **.PIF**. Note that Windows supports other extensions as well,

> **Note:** If you want the shortcut to appear on the Windows desktop, you should include the full path to one of the desktop directory in the command. In most Windows configurations, that directory can be referenced symbolically as **%userprofile\Desktop**. Some Windows versions also include an **All Users\Desktop** directory.

The ***mode*** `parameter` determines how Windows will display the application or folder when you run the shortcut. It must be **1** for a normal window, **2** for a minimized window (normally placed on the taskbar), or **3** for a maximized window.

The two (optional) parameters, ***iconfile*** and ***iconoffset*** allow you to specify the icon for the shortcut to use. (By default, SHORTCUT will use the default icon in the executable file.)

The final (optional) parameter ***hotkey*** specifies the keystroke which will call the shortcut. The keystroke should be entered in the same format as used in KEYSTACK 296; for example, **Ctrl-Alt-B**.

See Desktop Integration 10 for additional information on shortcuts use.

Display mode

If you provide a single parameter (a link file name), SHORTCUT will display the values for that link.

Examples

The command

```
     shortcut %_cmdspec /U  %@path[%_cmdspec] "Testing SHORTCUT"
"%UserProfile\Desktop\U%_cmdproc.lnk" 3
```

if executed from **4NT.EXE** creates a shortcut named ***U4NT*** in the current user's desktop. Executing the shortcut thus created starts **4NT.EXE** in its own directory, using Unicode output, in a maximized window. Moving the cursor over the shortcut on the desktop will display "Testing SHORTCUT'.

This is what happens if you now execute the command

```
shortcut "%UserProfile\Desktop\U%_cmdproc.lnk"
Command=C:\JPSOFT\RU\4nt.exe
Arguments=/U
Directory=C:\JPSOFT\RU\
Description=Testing SHORTCUT
Link=C:\Documents and Settings\ESF\Desktop\U4NT.lnk
Mode=3
```

7.108 SHRALIAS

Purpose: Retains global command history, directory history, alias and user function lists in memory when the command processor is not running.

Format: SHRALIAS [/U]

/U(nload) 362

Usage:

When you close all command processor sessions, the memory for the global command history, global directory history, global alias and global function lists is released. If you want the lists to be retained in memory even when the command processor is not running, you need to execute SHRALIAS.

The SHRALIAS command starts and initializes SHRALIAS.EXE, a small program which remains active and retains global lists when the command processor is not running. SHRALIAS.EXE must be stored in the same directory as the command processor or in a directory on your PATH. You cannot run SHRALIAS.EXE directly, it must be invoked internally by the SHRALIAS command.

Once SHRALIAS has been executed, the global lists will be retained in memory until you use SHRALIAS /U to unload the lists, or until you shut down your operating system.

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 the command processor running with global alias, global function, global command history, or global directory history enabled. If no global list is found, SHRALIAS will display an error.

If you start SHRALIAS from a temporary command processor session which exits after starting SHRALIAS, the command processor session may terminate and discard the shared lists before SHRALIAS can attach to them. In this case SHRALIAS.EXE will not be loaded. If you experience this problem, add a short delay with the DELAY 229 command after SHRALIAS is loaded and before your session exits.

SHRALIAS will not work in detached sessions (i.e., those started with DETACH 232, or with the AT utility), due to security issues within Windows. Therefore the SHRALIAS command is ignored for detached sessions.

For more information about global histories, function and alias lists, see Local and Global History Lists 52, Local and Global Functions 275, Local and Global Aliases 187.

Note: SHRALIAS shares the global lists (alias, history, directory history, and user functions) between both ***4NT*** and ***TC***.

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 the command processor exits unless SHRALIAS is loaded again before that time.

7.109 SMPP

Purpose: Send simple text (**SMS**) messages, typically to text-enabled cellular phones and similar devices.

Format: SMPP server username password recipient message

server SMS server name

username	User name for the SMS server
password	Password for the SMS server
recipient	Phone number or dotted IP of an SMS-enabled device
message	The message to send

See also: SENDMAIL 349, SNPP 363.

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.

Note: SMPP support is provided by **ipworks6.dll** and **ipwssl6.dll** (included in the ***4NT*** and ***TC*** distribution packages). Those files are installed by default in the command processor's directory.

7.110 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

7.111 SNPP

Purpose: Send messages to alphanumeric pagers.

Format SNPP server pagerid message

server	The SNPP server name
pagerid	The ID of the pager to receive the message
message	The message to send

See also: SENDMAIL 349, SMPP 362.

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.

Note: SNPP support is provided by **ipworks6.dll** and **ipwssl6.dll** (included in the ***4NT*** and ***TC*** distribution packages). Those files are installed by default in the command processor's directory.

7.112 START

Purpose: Start a program in another session or window.

Format: START ["title"] [/AFFINITY=n /ABOVENORMAL /BELOWNORMAL /HIGH /LOW /NORMAL /REALTIME /B /C /K /CM /Dpath /I /FS /INV /MAX /MIN /POS=x,y,width,height /L /LA /LD /LF /LH /RUNAS user password /SEPARATE /SHARED /SIZE=rows,cols /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 366	Priority	/LF 366	Local functions
/AFFINITY 366	Multiple CPUs	/LH 366	Local history list
/B 366	No new console	/LOW 366	priority
/BELOWNORMAL 366	Priority	/MAX 367	Maximized window
/C 366	Close when done	/MIN 367	Minimized window
/CM 366	Caveman	/NORMAL 367	Priority
/D 366	Startup directory	/PGM 367	Program name
/FS 366	Full screen window	/POS 367	Position of window
/HIGH 366	Priority	/REALTIME 367	Priority
/I 366	Inherit environment	/RUNAS 367	Run as other user
/INV 366	Invisible window	/SEPARATE 367	Separate session
/K 366	Keep when done	/SHARED 367	Shared session
/L 366	Local lists	/SIZE 367	Screen buffer size
/LA 366	Local aliases	/WAIT 367	For session to finish
/LD 366	Local directory history	/WIN 367	Windowed session

See also: DETACH 232.

Usage:

START is used to begin a new session, and optionally run a program in that session. If you use START with no parameters, it will begin a new command processor session. If you add a ***command***, START will begin a new session or window and execute that command.

START will return to the command processor prompt immediately (or continue a batch file), without waiting for the program to complete, unless you use /WAIT 367, or (***TC*** only) /CM 366.

If ***title*** is included, will appear on the task list and **Alt-Tab** displays. instead of the program name, which is the default. ***Title*** must be enclosed in double quotes, and cannot exceed 127 characters.

START always assumes that the first quoted string on the command line is the ***title*** . If there is a second quoted string it is assumed to be the ***command***. As a result, if the name of the program you are starting contains white space (and must therefore be quoted), and you don't specify a ***title,*** START will interpret the first quoted string as the ***title***, not the ***command***. To address this, use the /PGM 367 switch to indicate explicitly that the quoted string is the program name, or include a title before the program name. For example, to start the program *C:\Program Files\Proc.Exe* you could use either of the first two commands below, but the third command would not work:

Valid
```
   start /PGM "C:\Program Files\Proc.Exe"
   start "test" "C:\Program Files\Proc.Exe"
```
Invalid
```
   start "C:\Program Files\Proc.Exe"
```

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

START offers a large number of switches to control the session you start. In most cases you need only a few switches to accomplish what you want. The list below summarizes the most commonly used START options, and how you can use them to control the way a session is started.

Window controls: /FS 366, /MAX 367, /MIN 367, and /POS 367 allow you to start a character-mode windowed session in full screen mode, a maximized window, a minimized window, or a window with a specified position and size, respectively. /INV 366 starts an invisible window. /B 366 starts the program in the ***4NT*** window. The default is /WIN 367, which permits Windows to choose the position and size of the non-maximized window. If you start a graphics mode program, only /MAX 367 and /POS 367 are effective, and the position and size information associated with /POS 367 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 186 command after the window has been opened.

Session priority: The options /ABOVENORMAL 366, /BELOWNORMAL 366, /HIGH 366, /LOW 366, /NORMAL 367 and /REALTIME 367 allow you to select the new session's priority.

Program controls.

If ***progname*** is in the "App Paths" registry, its associated "Path" value (if it exists) is inserted into the beginning of the PATH 399 in the environment inherited by the program.

If ***progname*** is the name of a directory instead of an executable program, the command processor will start Windows Explorer in the specified directory. Explorer must be in the PATH 399, the **%WINDIR** directory, or the **%WINDIR\SYSTEM** directory for this feature to work correctly.

Progname inherits the environment as it exists when START is executed, unless /I 366 is used to select the default environment.

It ***progname*** specifies either **4NT.EXE** or **TCMD.EXE**, the options /L 366, /LA 366, /LD 366, /LF 366 and /LH 366 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 366 option. **WARNING!** If ***progname*** is a DOS program, this option may not work properly. To

avoid such problems, you can change the default directory before START, and restore it afterwards.

If ***progname*** is a 16-bit Windows application, by default is starts in a shared virtual machine. You may use the /SEPARATE 367 option to force creation of a unique virtual machine.

When ***command*** is finished, /C 366 closes the session (the default for Windows sessions), while /K 366 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 422 internal variable.

Options:

/ABOVENORMAL	Set the priority above normal.
/AFFINITY=n	On multiple processor machines, set the processor affinity for this process. Acceptable values are 0 to ***n***-1 (where ***n*** is the number of available processors)
/BELOWNORMAL	Set the priority below normal.
/B	***(4NT)*** The program is started without creating a new window or console, *i.e.* in the ***4NT*** window. Normally, the application is started in its own window. For compatibility with CMD.EXE, **/B** also disables **Ctrl-C** processing for the program.
/C	Close the session or window when the application ends.
/CM	***(TC*** with Caveman 86***)*** Run a character-mode application under Caveman 86. Use this option to force an application to run under Caveman 86 even if Caveman 86 is not specifically enabled for that application. For more details see Console Applications and the Console Window 86. **/CM** will be ignored if Caveman 86 is not enabled via the Caveman Applications Dialog 158 (accessible via the **Options** menu). ***TC*** always waits for the application to finish when **/CM** is used, even if you do not use **/WAIT**.
/D	Specifies the startup directory. Include the directory name immediately after the **/D**, with no intervening spaces or punctuation. Due to limitations in the way Windows starts DOS programs, **/D** is ignored when starting DOS applications.
/FS	Start the console application in full-screen mode.
/HIGH	Start the window at high priority.
/I	Inherit the default environment, if any, rather than the current environment.
/INV	Start the session or window as invisible. No icon will appear and the session will only be accessible through the Task Manager or Window List.
/K	(Keep session or window at end) The session or window continues after the application program ends. Use the EXIT 262 command to end the session.
/L	Start the command processor with local alias, function, history and directory history lists. This option is equivalent to specifying all of /LA 366, /LD 366, /LF 366, and /LH 366 (below).
/LA	Start the command processor with a local alias list. See ALIAS 187 for information on local and global alias lists.
/LD	Start the command processor with a local directory history list. See Local and Global History Lists 52 for information on local and global directory history lists.
/LF	Start the command processor with a local function list. See FUNCTION 275 for information on local and global function lists.
/LH	Start the command processor with a local history list. See Local and Global History Lists 52 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.
/NORMAL	Start the window at normal priority.
/PGM	The quoted string following this option is the program name. Any additional text beyond the quoted string is passed to the program as its parameters, so to use other START switches you must place them before **/PGM** which must be the last option for START. You can use **/PGM** to allow START to differentiate between a quoted long filename and a quoted title for the session.
/POS=*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.
/SEPARATE	Start a 16-bit Windows application in a separate virtual machine. Normally, all 16-bit Windows applications are started in the same virtual machine, see /SHARED 367.
/SHARED:	Start a 16-bit Windows application in the shared virtual machine (default). See also /SEPARATE 367. Included only for compatibility with CMD.EXE.
/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.)
/WAIT	Wait for the new session or window to finish before continuing.
/WIN	Start the new console session as a window (this is the default.) See also /FS 366 and /B 366.

7.113 SWITCH

Purpose: Select commands to execute in a batch file based on a value.

Format:
```
SWITCH expression
CASE value1 [.OR. value2 [.OR. value3 …]]
    [commands]
CASE value4
    [commands]
[DEFAULT
    commands]
ENDSWITCH
```

expression	An environment variable, internal variable, variable function, text string, or a combination of these elements, that is used to select a group of commands.
value1, value2	A value to test or multiple values connected with **.OR.**
commands	One or more commands to execute if the expression matches the value. If you use multiple commands, they must be separated by command separators or placed on separate lines of a batch file.

See also: IF 286 and IFF 287.

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***.

The command processor 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 54 for details. If a match is found, the ***commands*** following the matched CASE statement are executed, and the batch file continues with the commands that follow ENDSWITCH. If there are any matches in subsequent CASE statements, they are ignored.

If during the search for a match the DEFAULT statement is encountered, the ***commands***, if any, following it are executed, and the batch file continues with the commands that follow ENDSWITCH. Any CASE statements after the DEFAULT statement are ignored.

SWITCH commands can be nested.

You can exit from all SWITCH / ENDSWITCH processing by using GOTO 282 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
```

7.114 SYNC

Purpose: Synchronize two directories

Format: SYNC [/A:... /C /D /E /F /G /J /K /L /M /N[est] /O /P /Q /R /S[n] /T /U /V /W /X] dir1 dir2

dir1	First directory (and source for a /W)
dir2	Second directory (and target for a /W)

/A:221...	Attribute switch	/M222	Modified files (not Archived)
/C222.	Changed source files	/N222	Disable
/D222	Copy encrypted files	/O223	Only if no target file
/E222	No error messages	/P223	Prompt
/F222	No empty subdirectories	/Q223	Quiet
/G222	Display percentage completed	/R223	Replace
/H222	H(idden included)	/S223	Subdirectories included
/I"text"222	Match description	/T223	Totals
/J222	Restartable copy	/V223	Verify
/K222	Keep RDONLY attribute	/W371	Delete non-matching target
/L222	ASCII-mode FTP transfer		

See also: COPY 216 and MOVE 308.

File Selection

Supports extended wildcards 19 and ranges 22.

Internet: Can be used with FTP servers 42.

Usage:

SYNC will synchronize two directories, copying the updated files from each directory to the other.

Options:

/A:... Select only those files that have the specified attribute(s) set. See Attribute Switches 28 for information on the attributes which can follow /A:. See the cautionary note under **Advanced Features** above before using ***/A:*** when both ***dir1*** and ***dir2*** contain file descriptions. Do not use /A: with ***@file*** lists. See @file lists 32 for details. Hidden or

system files selected by this option overwrite hidden or system files in the target directory.

/C Copy files only if the destination file exists and is older than the source file. This option is useful for updating the files in one directory from those in another without copying any files not already in the target directory. Do not use /C with ***@file*** lists. See @file lists 32 for details.

/D ***(***Windows XP+ Only***)*** Force copy of an encrypted file even when the target will be decrypted.

/E Suppress all non-fatal error messages, such as **File not found** or **Can't copy file to itself**. Fatal error messages, such as **Drive not ready**, will still be displayed. This option is most useful in batch files and aliases.

/F When used with **/S**, SYNC will not create any empty subdirectories.

/G Displays the percentage copied and the transfer rate (in Kbytes/second). Useful when copying large files across a network or via FTP to ensure the copy is proceeding. When /V 223 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 28 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 223 switch, or use ATTRIB 198. Do not use /M with ***@file*** lists. See @file lists 32 for details.

/N Do everything except actually perform the copy. This option is useful for testing the result of a complex SYNC command. /N displays how many files would be copied. /N does not prevent creation of destination subdirectories when it is used with /S 223.

A **/N** with one of the following arguments has an alternate meaning:

- **e** Don't display errors.
- **s** Don't display the summary.
- **t** Don't update the CD / CDD extended directory search 14 database *(JPSTREE.IDX)*.

/P Ask the user to confirm each source file. Your options at the prompt are explained in detail under Page and File Prompts 41. See also: the /Q 223 option below.

/Q Don't display filenames, percentage copied, total number of files copied, etc... When used in combination with the /P 223 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 223.

/R Prompt the user before overwriting an existing file. Your options at the prompt are explained in detail under Page and File Prompts 41.

/S Copy the subdirectory tree starting with the files in the source directory plus each subdirectory below that. If the destination subdirectories don't exist, SYNC will attempt to create them. If SYNC /S creates one or more destination directories, they will be added automatically to the extended directory search 14 database.

If you attempt to use SYNC /S to copy a subdirectory tree into part of itself, SYNC will detect the resulting infinite loop, display an error message and exit. Do not use /S with ***@file*** lists. See @file lists 32 for details.

If you specify a number after the /S, SYNC will limit the subdirectory recursion to that number. For example, if you have a directory tree "\a\b\c\d\e", /S2 will only affect the "a", "b", and "c" directories.

/T Turns off the display of filenames, like /Q 223, 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***.

7.115 TAIL

Purpose: Display the end of the specified file(s).

Format: TAIL [range ... [/I 373"text"]] [/A: 373[attrlist] /C 373nn /F 373 /N+ 373x /N 373[ ]n /P 373 /Q 373 /V 373] {@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 32 for details).

/A: 373 (Attribute select)
/C 373 (number of bytes)
/F 373(ollow)
/I 373"*text*" (description range)
/N+ 373*x* (skip *x* lines before display)
/N 373(umber of lines)
/P 373(ause)
/Q 373(uiet)
/V 373(erbose)

See also: HEAD 283, LIST 298, and TYPE 384.

File Selection

Supports extended wildcards 19, ranges 22, multiple file names 29, and include lists 30.

Internet: Can be used with FTP servers 42, including HTTP/HTTPS files, e.g.

```
tail "http://jpsoft.com/notfound.htm"
```

Usage:

The TAIL command displays the last part of a file or files. It is normally only useful for displaying ASCII text files (i.e. alphanumeric characters arranged in lines separated by CR/LF). Executable files (*.COM* and *.EXE*) and many data files may be unreadable when displayed with TAIL because they include non-alphanumeric characters or unusual line separators.

You can press **Ctrl-S** to pause TAIL's display and then any key to continue.

The following example displays the last 15 lines of the files *MEMO1* and *MEMO2*:

```
tail /n15 memo1 memo2
```

To display text from the clipboard use **CLIP:** as the file name. **CLIP:** will not return any data if the clipboard does not contain text. See Highlighting and Copying Text 83 for additional information on **CLIP:**.

- ***FTP Usage***

TAIL can also display files on FTP servers 42. For example:

```
tail "ftp://jpsoft.com/index"
```

You can also use the IFTP 288 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 39. 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 the 1002-nd line of *xxx*

`set x=%@execstr[tail /n 2 xxx]` sets **x** to the contents of the penultimate line of *xxx*

Options:

/A:*[attributelist]*
Select only those files that match the specified attribute(s).See Attribute Switches 28 for details.

/Cnn*[b|k|m]*
Display ***nn*** bytes, 512-byte ***b***locks, ***k***ilobytes, or ***m***egabytes, when there is no suffix, for suffix b, suffix k, or suffix m, respectively.

/F Continuously monitor the file and display new lines until the command is interrupted, e.g, using ***Ctrl-C*** or Ctrl-***Break***..

/I"*text*"
Select files by a descriptor range 28. 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 373 is also specified.

/N+x Skip ***x*** lines from the beginning of the file, than start displaying lines. If the **/N+** option is specified without specifying ***x***, the option is ignored. This option does not affect the number of lines displayed (unless the start line is too close to the end of file)

Example: `TAIL /N+5 file` will display 10 lines (the default) after skipping 5 lines.

/P Pause and prompt after displaying each page.

/Q Do not display a header for each file. This is the default behavior, but an explicit /Q may be needed to override an alias that forces /V 373.

/V Display a header for each file.

7.116 TASKEND

Purpose: End the specified process.

Format: TASKEND [/F] pid | name | "title"

pid The process ID
name The process name
title Window title

/F(orce) 374

See also:: TASKLIST 374, _PID 420, _DETACHPID 415, _WINTITLE 424

Usage:

Windows applications (and Windows itself) run as one or more processes or tasks. You can use the TASKLIST command to display a list of currently-running tasks. TASKEND can be used to end a task.

When you use TASKEND, you must specify the task you want to end by process ID number, by name

(usually the name of the executable file that started the task) or by window title. If you use the Window title to specify the task, you must enclose it in double quotes. You can use wild cards and extended wildcards in the window title.

If you use TASKEND without the **/F** option, the effect is much the same as closing a window by clicking the close button. The application is notified of the request to end the task and has an opportunity to save data, prompt whether you mean to shut down, and perform other normal "close" operations.

If you use the **/F** option with TASKEND, the application is shut down abruptly and has no chance to save data. Use of the **/F** option is only recommended for unusual circumstance and advanced users because of the possibility of data loss.

Using this command may require DEBUG privilege.

Option:

/F Forces the task or application to end immediately, with no opportunity to save data, prompt the user, etc. Use this option with caution; it can possibly lead to system instability and data loss or corruption.

7.117 TASKLIST

Purpose: Display a processes list

Format: TASKLIST [/O /P] [name]

name Process name or window title

/O(rder by PID) 374 /P(ause) 374

See also:: TASKEND 373.

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 limit the output of TASKLIST by specifying the task name that you wish to see. The name can contain wildcards and extended wildcards 19.

Options:

/O Sort the output by Process ID (PID).

/P Wait for a key to be pressed after each screen page before continuing the display.

7.118 TCTOOLBAR

Purpose: Change the tool bar buttons.

Format: TCTOOLBAR [/C /U /R filename] button [, flags, text, command]

button The button number (1 – 32)

flags	0=Echo, 1=Echo & Execute, 2=Execute without echo
text	The button text
command	The command line to execute.

/C(lear) 375 /U(pdate) 375
/R(ead file) 375

Usage:

TCTOOLBAR lets you configure the ***TC*** tool bar buttons (you can also use the Configure Tool Bar dialog 79 available from the Options 77 menu). The changes you make can be temporary or, with the **/U** option, written to the **TCMD32.INI** file so that they will be loaded the next time ***TC*** starts.

There are a maximum of 32 buttons on the tool bar. The ***button*** parameter must be a number from 1 to 32 to select the button you want to work with. If you enter a command like

```
tctoolbar 1
```

The button with that number, if it is currently visible, will be removed from the tool bar. If you want to add or modify a button, you must include the ***flags***, ***header***, and ***command*** parameters in the command.

The ***flags*** parameter specifies what happens when you click the button. If it is 0, the button's command is added to the command line at the current cursor position, but the command line is not executed immediately. You can then edit the command, add additional options and parameters, etc., before you press Enter to execute it (or Esc to cancel the action). If the ***flags*** parameter is set to 1, the command text is added to the command line and then the entire command line is immediately executed. If it is 2, ***TC*** adds the button's command to its in-memory copy of the command line and then executes that, without actually displaying the text.

The ***text*** parameter specifies the text that appears on the button. If the text contains white space or other special characters, it must be enclosed in double quotes.

The ***command*** parameter contains the text that is placed on the command line and / or executed when the button is clicked. It may be any internal or external command, an alias name, batch file name, or any other text that you want added to the command line.

Option:

/C Clear all entries from the toolbar.

/R Load the toolbar button definitions from the specified file. **/R** will **not** clear an existing toolbar; you must use **/C** for that. The file should be in the same format as the **[Buttons]** section in TCMD32.INI:

Bn=flags,text,command

n - the button number (1 - 32)
flags - 0-echo, 1=echo and execute, 2=execute without echo
text - the text to display on the button
command - the command to execute

/U Write the changed button definition to the **TCMD32.INI** file so that it will be included the next time ***TC*** starts.

7.119 TEE

Purpose: Copy standard input to both standard output and a file.

Format: TEE [/A /D /T] file...

file One or more files that will receive the "tee-d" output.

/A(ppend) 376 /T(ime) 376
/D(ate) 376

See also: Y 395, piping 39 and redirection 36.

Usage:

TEE is normally used to "split" the output of a program so that you can see it on the display and also save it in a file. It can also be used to capture intermediate output before the data is altered by another program or command.

TEE gets its input from standard input (usually the piped output of another command or program), and sends out two copies: one to standard output, the other to the ***file(s)*** that you specify. TEE is not likely to be useful with programs which do not use standard output, because these programs cannot send output through a pipe.

For example, to search the file *DOC* for any lines containing the string ***Take Command***, make a copy of the matching lines in *TC.DAT*, sort the lines, and write them to the output file *TCS.DAT*:

```
ffind /t"Take Command" doc | tee tc.dat | sort > tcs.dat
```

If you are typing at the keyboard to produce the input for TEE, you must enter a **Ctrl-Z** to terminate the input.

See Piping 39 for more information on pipes.

Option:

/A Append to the file(s) rather than overwriting them.

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

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

7.120 TEXT

Purpose: Display a block of text in a batch file.

Format: TEXT
.
.
.
ENDTEXT

See also: ECHO 253, ECHOS 255, SCREEN 342, SCRPUT 344, and VSCRPUT 391.

Usage:

TEXT can only be used in batch files. Both TEXT and ENDTEXT must be entered as the only commands on their respective lines.

The TEXT command is useful for displaying menus, tables, special characters, or multiline messages. TEXT will display all lines in the batch file between itself and the terminating ENDTEXT. The display starts at the current display position, which allows you to start its display with other text, e.g., from the ECHOS 255 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 215 or the COLOR 216 command to set the default screen colors before executing TEXT.

Redirecting TEXT output

To redirect or pipe the entire block of text, use redirection 36 or piping 39 on the TEXT command itself as shown in the Examples 377 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 315 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
```

7.121 TIME

Purpose: Display or set the current system time.

Format: TIME [/S [server] /T] [*hh* [:mm:ss]]] [AM | PM]

hh The hour (0 - 23)
mm The minute (0 - 59)
ss The second (0 - 59)

/S(erver time) 378 /T (Display only) 379

See also: DATE 224.

Usage:

If you don't enter any parameters, TIME will display the current system time and prompt you for a new time. Press Enter if you don't wish to change the time; otherwise, enter the new time:

```
[c:\] time
Thu  Sep 30, 2004  9:30:06
Enter new date (mm-dd-yy):
```

TIME defaults to 24-hour format, but you can optionally enter the time in 12-hour format by appending **a**, **am**, **p**, or **pm** to the time you enter. For example, to enter the time as 9:30 am:

```
time 9:30 am
```

The operating system adds the system time and date to the directory entry for every file you create, modify, or access. If you keep both the time and date accurate, you will have a record of when you last updated each file.

Options:

/S *server* Sets the date and time from the specified internet time server. If no server is specified, TIME uses the server defined in the TimeServer 147 .INI file entry (the default is `clock.psu.edu`).

/T Displays the current time but does not prompt you for a new time. You cannot specify a new time on the command line with **/T**. If you do, the new time will be ignored.

7.122 TIMER

Purpose: TIMER is a system stopwatch.

Format: TIMER [ON | OFF | /Q /S /1 /2 /3]

ON Force the stopwatch to reset and start
OFF Force the stopwatch to stop

/1 stopwatch #1 (default) /Q quiet
/2 stopwatch #2 /S split
/3 stopwatch #3

Usage:

The TIMER command accepts its parameters in any order, and acts on the specified one of three possible timers (system stopwatches) by turning it on or off, or by displaying its current elapsed time. The TIMER command with neither of the keywords **ON** and **OFF** nor the **/S** option toggles the state of the timer.

If you execute TIMER or TIMER /S when the timer is off, or execute TIMER ON at any time, the current time of day is displayed, and the stopwatch starts from :

```
[c:\] timer
Timer 1 on:  12:21:46
```

If you execute TIMER /S when the timer is on, the elapsed time is displayed:

```
[c:\] timer /s
Timer 1  Elapsed time: 0:00:12.06
```

If you execute TIMER when it is on, or execute TIMER OFF, the stopwatch stops, the current time and the elapsed time are displayed, and the elapsed time is reset:

```
[c:\] timer
Timer 1 off:  12:21:58
Elapsed time: 0:00:12.06
```

There are three stopwatches available (1, 2, and 3) so you can time multiple overlapping events. By default, TIMER uses stopwatch #1.

TIMER is particularly useful for timing events in batch files. For example, to time both an entire batch file, and an intermediate section of the same file, you could use commands like this:

```
rem Turn on timer 1
timer
rem Do some work here
rem Turn timer 2 on to time the next section
timer /2
rem Do some more work
echo Intermediate section completed
rem Display time taken in intermediate section
```

```
timer /2
rem Do some more work
rem Now display the total time
timer
```

The smallest interval TIMER can measure depends on the operating system you are using, your hardware, and the interaction between the two. However, it should never be more than 60 ms.

You can also retrieve the elapsed time of a timer using the @TIMER[] 492 function.

Options:

/1 Use timer #1 (the default).

/2 Use timer #2.

/3 Use timer #3.

/Q Don't display any messages.

/S Display a split time without stopping the timer. To display the current elapsed time but leave the timer running:

```
[c:\] timer /s
Timer 1 elapsed: 0:06:40.63
```

ON Start the timer regardless of its previous state (on or off). Otherwise the TIMER command toggles the timer state (unless **/S** is used).

OFF Stops the timer.

7.123 TITLE

Purpose: Change the window title.

Format: TITLE [/P] title

/P(rompt characters) 381

title The new window title.

See also: the TITLEPROMPT 400 variable and the ACTIVATE 186 and WINDOW 392 commands.

Usage:

TITLE changes the text that appears in the caption bar at the top of the command processor window. You can also change the window title with the WINDOW command or the ACTIVATE command.

The title text should not be enclosed in quotes unless you want the quotes to appear as part of the actual title.

To change the title of the current window to "Title Test":

```
title Title Test
```

Options:

/P Support the special characters in PROMPT 329.

7.124 TOUCH

Purpose: Change a file's time stamps 525, and optionally create a file.

Format: TOUCH [/A:[[-][+]rhsdaecjot] /C [/D[acw][date] /E /F /I""text"" /N /Q /R[:acw]file /Sn /T[acw][hh:mm[:ss[.dd]]] file...

file One or more files whose date and/or time stamps are to be changed.

/A 381 Attribute select :	/N 382 No action
/C 381 Create file	/Q 382 Quiet
/D 381 Date	/R 382 Reference file
/E 382 No error messages	/S 382 Subdirectories
/F 382 Force read-only files	/T 382 Time
/I 382 Match descriptions	

File Selection:

Supports attribute switches 28, extended wildcards 19, ranges 22, multiple file names 29, subdirectories, catalog files 32, and include lists 30.

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 381: option to include directories. /A:D will select directories only.

Options:

/A: Select only those files that have the specified attribute(s) set. See Attribute Switches 28 for information on the attributes which can follow /A:.

/C Create ***file*** (as a zero-byte file) if it does not already exist. You cannot use wildcards with /C, but you can create multiple ***files*** by listing them individually on the command line.

/D If neither /R 382 nor /D 381 are specified, the current date is used. If the /D 381 option is specified without date, TOUCH will not modify the date even if /R 382 is also specified. If the /D 381 option is followed by date, and /R 382 is not specified, date is used. Date must

not be quoted. If both /R 382 and /D 381 with date are specified, the one specified later in the command takes effect.

/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 382 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 28 for details.

/N Display what would occur without actually doing it.

/Q Do not display normal messages.

/R The /R option permits duplication of the timestamp of ***ref_file***. For example, if you recompile an old program (e.g., to obtain an intermediate file that has long been deleted) you may want to use the timestamp of the source file that was last changed as the timestamp of the newly built duplicate of the original object file to prevent a "make" from attempting to rebuild everything else in the project as shown in the example:

```
touch /r project.c project.obj
```

Another use could be to synchronize files without rendering the current version inaccessible during the synchronization:

```
touch /c /r c:\jpsoft\jphelp.chm %temp\jphelp.chm
copy /u ftp://jpsoft.com/help/jphelp.chm %temp\jphelp.chm
```

In the above example TOUCH creates an empty file with the timestamp of your already existing help file; COPY 216 updates the empty file if a newer version is available (beware of timestamp synchronization across the Internet!).

/S TOUCH all matching files in the specified directory and its subdirectories. Do not use /S with @file lists. See @file lists 32 for details.

If you specify a number after the /S, TOUCH will limit the subdirectory recursion to that number. For example, if you have a directory tree "\a\b\c\d\e", /S2 will only affect the "a", "b", and "c" directories.

/T If neither /R 382 nor /T 382 are specified, the current time is used. If the /T 382 option is specified without time, TOUCH will not modify the time even if /R 382 is also specified. If the /T 382 option is followed by time, and /R 382 is not specified, time is used. (Time must not be quoted). If both /R 382 and /T 382 with time are specified, the one specified later in the command takes effect.

7.125 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).

7.126 TREE

Purpose: Display a graphical directory tree.

Format: TREE [[/A:[[-|+]rhsadecijopt /A /B /D /F /H /P /S[n] /T[:a|c|w] /Z] dir...

dir The directory to use as the start of the tree. If one or more directories are specified, TREE will display a tree for each specified directory. If none are specified, the tree for the current working directory is displayed.

/A: (Attribute select)
/A (SCII)
/B (are)
/D (escriptions)
/F (iles)
/H (idden directories)
/P (ause)
/S (file size)
/T (ime and date)
/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 the command processor to use a font which can display standard extended ASCII characters.

You can print the display, save it in a file, or view it with LIST 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.

Options:

/A Display the tree using standard ASCII characters. You can use this option if you want to save the directory tree in a file for further processing or print the tree on a printer which does not support the graphical symbols that TREE normally uses.

/A:[..] Select only those files that match the specified attribute(s).See Attribute Switches for details.

/B Display the full pathname of each directory, without any of the line-drawing characters.

/D Display file and directory descriptions.

/F Display files as well as directories. If you use this option, the name of each file is displayed beneath the name of the directory in which it resides.

/H Display hidden as well as normal directories. If you combine **/H** and /F 384, hidden files are also displayed.

/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 41.

/S If you specify a number after the /S, TREE will limit the subdirectory recursion to that number. For example, if you have a directory tree "\a\b\c\d\e", /S2 will only affect the "a", "b", and "c" directories.

/T Display the time and date for each directory. If you combine **/T** and /F 384, 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 572 and FAT32 564 volumes).
/T:c Creation date and time.
/T:w Last modification ("*write*") date and time (default).

/Z Display the size of each file. This option is only useful when combined with /F 384.

7.127 TRUENAME

Purpose: Find the full, true path and file name for a file

Format: TRUENAME file

See also: The @TRUENAME 492 variable function.

Usage:

Network reassignments, junctions, and the SUBST command can obscure the true name of a file. TRUENAME "sees through" these obstacles and reports the fully qualified name of a file.

The following example uses TRUENAME to get the true pathname for a file:

```
[c:\] subst d: c:\util\test
[c:\] truename d:\test.exe
c:\util\test\test.exe
```

7.128 TYPE

Purpose: Display the contents of the specified file(s).

Format: TYPE [/A:[[-][+]rhsadecijopt] /I"text" /L /P] [@*file*] *file*...

file The file or list of files that you want to display.
@file A text file containing the names of the files to display, one per line (see @file lists 32 for details).

/A: (Attribute select) 385 /L(ine numbers) 386
/I"text" (match description) 386 /P(ause) 386

See also: HEAD 283, TAIL 371, LIST 298.

File Selection

Supports attribute switches 28, extended wildcards 19, ranges 22, multiple file names 29, and include lists 30.

Internet: Can be used with FTP and HTTP servers 42, e.g.

```
type "http://jpsoft.com/notfound.htm"
```

Usage:

The TYPE command displays a file. It is normally only useful for displaying text files (i.e. alphanumeric characters arranged in lines separated by CR/LF). Most text files use either ASCII or Unicode.

Executable files (*.COM* and *.EXE*) and many data files may be unreadable when displayed with TYPE because they include non-alphanumeric characters or unusual line separators.

To display the files *MEMO1* and *MEMO2*:

```
type /p memo1 memo2
```

You can press **Ctrl-S** to pause TYPE's display and then any key to continue.

To display text from the clipboard use **CLIP:** as the file name. CLIP: will not return any data if the clipboard does not contain text. See Redirection 36 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 36 is useful if you want to add line numbers to a file, for example:

```
type /l myfile > myfile.num
```

- **NTFS File Streams**

TYPE supports file streams on NTFS drives. You can type an individual stream by specifying the stream name, for example:

```
type streamfile:s1
```

See NTFS File Streams 526 for additional details.

Options:

/A: Select only those files that have the specified attribute(s) set. See Attribute Switches 28 for information on the attributes which can follow **/A:**. Do not use /A: with @file lists. See @file lists 32 for details.

/I"*text*" Select files by matching text in their descriptions. The text can include wildcards 19 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 32 for details.

/L Display a line number preceding each line of text.

/P Prompt after displaying each page. Your options at the prompt are explained in detail under Page and File Prompts 41.

7.129 UNALIAS

Purpose: Remove aliases from the alias list.

Format: UNALIAS [/Q /R *file*...] *alias*...
or
UNALIAS *

alias One or more aliases to remove from memory.
file One or more files from which to read the aliases to be undefined.

/Q(uiet) 387 /R(ead file) 387

See also: ALIAS 187 and ESET 258.

Usage:

The command processor maintains a list of the aliases that you have defined. The UNALIAS command will remove aliases from that list. UNALIAS supports wildcards in the alias name.

For example, to remove the alias DDIR:

```
unalias ddir
```

To remove all the aliases:

```
unalias *
```

To remove all the aliases that begin with "DD":

```
unalias dd*
```

If you keep aliases in a file that can be loaded with the ALIAS /R 187 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.

(TC) You can also delete individual aliases with the Alias dialog 81.

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 187 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 571 if no filename is present and input is redirected.

7.130 UNFUNCTION

Purpose: Remove user-defined functions from the function list.

Format: UNFUNCTION [/Q /R *file*...] *function*...
or
UNFUNCTION *

function One or more functions to remove from memory.
file One or more files from which to read functions to be undefined.

/Q(uiet) 388 /R(ead file) 388

See also: FUNCTION 275 and ESET 258.

Usage:

The command processor maintains a list of the functions that you have defined. The UNFUNCTION command will remove functions from that list. UNFUNCTION supports wildcards in the function name.

To remove the function DDIR:

```
unfunction ddir
```

To remove all the functions:

```
unfunction *
```

To remove all the functions that begin with "DD":

```
unfunction dd*
```

If you keep functions in a file that can be loaded with the FUNCTION /R 275 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 *.

(TC) You can also delete individual functions from the Functions Dialog 82.

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 275 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 571 if no filename is present and input is redirected.

7.131 UNSET

Purpose: Remove variables from the environment or the registry.

Format: UNSET [/D /E /Q /S /U /V /R file...] name [name...]}]
or
UNSET *******

name One or more variables to removed (wildcards accepted except for registry variables).
file One or more files from which to read variables to be removed.

/D(efault) 389
/E(nvironment) 389
/Q(uiet) 389
/R(ead) 389
/S(ystem) 389
/U(ser) 389
/V(olatile) 389

See also: ESET 258 and SET 351.

Usage:

UNSET removes one or more variables from the environment or from the Windows Registry.

For example, to remove the environment variable **CMDLINE**:

```
unset cmdline
```

If you use the command **UNSET ***, all of the environment variables will be deleted:

```
unset *
```

(TC) You can also remove individual variables from the environment with the Environment dialog 81.

UNSET can be used in a batch file, in conjunction with the SETLOCAL 358 and ENDLOCAL 256 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 351 command and the general discussion of the environment 395.

Note: You cannot use UNSET with GOSUB variables 280.

Use caution when removing environment variables, and especially when using **UNSET ***. Many programs will not work properly without certain environment variables; for example, the command processor depends on PATH.

Registry Variables: Default, System, User, and Volatile registry variables can be manipulated with the UNSET command's /D 389, /S 389, /U 389 and /V 389 switches, respectively. To remove the variable from both the registry and from the local environment, use both the /E 389 switch and the registry variable selection switch together. (You cannot use wildcards for the variable name.) For example, to remove the volatile variable ***myvar*** from both the registry and the local environment, use:

```
unset /v /e myvar
```

Use caution when directly removing registry variables as they may be essential to various Windows processes and applications.

Options:

/D Delete a default variable from the registry (HKCU\.DEFAULT\Environment).

/E When used together with one of /D 353, /S 354, /U 354, or /V 354, unsets both the registry variable and the local environment variable.

/Q Prevents UNSET from displaying an error message if one or more of the variables do not exist. This option is most useful in batch files, for removing a group of variables when some of the variables may not have been defined.

/R Read environment variables to be UNSET from a file. This is much faster than using multiple UNSET commands in a batch file, and can be more selective than **UNSET ***. The file format may be the same as that used by the SET 351 /R command (see SET 351 for more details), or it could just be one variable per line, wildcards not processed.

UNSET /R will read from STDIN 571 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)

7.132 VER

Purpose: Display the command processor and operating system versions.

Format: VER [/R]

/R(evision) 390

Usage:

Version numbers consist of a one-digit major version number, a separator, and a one- or two-digit

minor version number. VER uses the default decimal separator defined by the current country information. The VER command displays version numbers for both ***4NT*** / ***TC*** and Windows:

```
[c:\] ver
4NT 8.0   Windows XP 5.1
```

Option:

/R Display the command processor and operating system internal revision level (if any), plus your command processor registered name.

7.133 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. The command processor supports VERIFY as a "do-nothing" command, for compatibility with CMD.EXE. This avoids **unknown command** errors in old batch files which use the VERIFY command.

If used without any parameters, VERIFY will display the state of the verify flag:

```
[c:\] verify
VERIFY is ON
```

7.134 VOL

Purpose: Display disk volume label(s).

Format: VOL [*d:*] ...

d: The drive or drives to search.

Usage:

Each disk may have a volume label, created when the disk is formatted or with the external LABEL command. Also, every disk formatted with Windows has a volume serial number.

The VOL command will display the volume label and, if available, the volume serial number of a disk volume. If the disk doesn't have a volume label, VOL will report that it is "unlabeled." If you don't specify a drive, VOL displays information about the current drive:

```
[c:\] vol
Volume in drive C: is MYHARDDISK
```

If available, the volume serial number will appear after the drive label or name.

To display the disk labels for drives A and B

```
[c:\] vol a: b:
Volume in drive A: is unlabeled
Volume in drive B: is BACKUP_2
```

VOL will also return volume information for UNC names.

See also: @LABEL 476.

7.135 VSCRPUT

Purpose: Display text vertically in the specified color.

Format: VSCRPUT row col [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

See also: SCRPUT 344.

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 552 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.

(TC) The maximum ***row*** value is determined by the current height of the ***TC*** window. The maximum ***column*** value is determined by the current virtual screen width (see Resizing the Take Command Window 84 for more information).

You can also specify the *row* and *column* as offsets from the current cursor position. Begin the value with a plus sign [**+**] to move down the specified number of rows or to the right the specified number of columns before displaying text, or with a minus sign [**-**] to move up or to the left.

If you specify 999 for the ***row***, VSCRPUT will center the text vertically. If you specify 999 for the ***column***, VSCRPUT will center the text horizontally.

VSCRPUT does not move the cursor when it displays the ***text***.

The following batch file fragment displays an X and Y axis and labels them:

```
cls bright white on blue
drawhline 20 10 40 1 bright white on blue
drawvline 2 10 19 1 bright white on blue
scrput 21 20 bright red on blue X axis
vscrput 8 9 bright red on blue Y axis
```

7.136 WHICH

Purpose: Display the command type and what it would execute.

Format: WHICH [/A] command [command ...]

command One or more commands or files.

/A 392(ll)

Usage:

WHICH displays information about internal and external commands, Aliases 160 (including keystroke aliases), and files. When a file matches an applicable Executable Extension 33 or Windows File Association 535, that data will be displayed. The exact information reported depends on the type of command or file you specify. For example:

```
[c:\] which cdd buildtree notepad test.btm test.exe test.xyz test.doc
donothing
CDD is an internal command
buildtree is an alias : cdd /s
notepad is an external: C:\windows\notepad.exe
test.btm is a batch file : C:\test.btm
test.exe is an external : C:\test.exe
test.xyz is an executable extension : C:\path\mybatch.btm C:\test.xyz
test.doc is associated with : C:\Program Files\Microsoft
Office\OFFICE11\WINWORD.EXE
donothing is an unknown command
```

If the command is an abbreviated alias, WHICH will display the full name; i.e.:

```
[c:\] alias opt*ions=*option
[c:\] which opt
opt*ions is an alias : *option
```

WHICH can also recognize Plugin 325 commands, REXX 175 files, EXTPROC 176 files, and associated files.

Note: WHICH does not support wildcard specifications. 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 399.

See Executable Files and File Searches 533 for details on the order in which various locations are searched.

See also: @SEARCH 488, ASSOC 197, FTYPE 274.

Option:

/A Display all matching commands. (Normally WHICH only displays the first match.)

7.137 WINDOW

Purpose: Minimize or maximize the current window, restore the default window size, or change the window title.

Format: WINDOW [MAX | MIN | RESTORE | HIDE | TRAY | FS | WIN | TOPMOST | NOTOPMOST | TOP | BOTTOM | /POS=left,top,width,height | /SIZE=rows,columns | /TRANS=n | "newtitle"]

newtitle	A new title for the window
height	New height of window
width	New width of window
left	New position of the left border of window
top	New position of the top border window
rows	New height of window
columns	New width of window

/POS(ition) /SIZE (of screen buffer)

See also: ACTIVATE and TITLE.

Usage:

WINDOW is used to control the appearance and title of the current (command processor) window. Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

4NT supports the **MAX**, **MIN**, **RESTORE**, **HIDE**, **TRAY**, **FS**, **WIN**, **/POS**, **/SIZE**, and ***newtitle*** options.

TC supports the **MAX**, **MIN**, **RESTORE**, **HIDE**, **TRAY**, **TOPMOST**, **NOTOPMOST**, **TOP**, **BOTTOM**, **/POS**, and ***newtitle*** options.

Note: You can specify only one **WINDOW** option at a time. The different options cannot be combined in a single **WINDOW** command. To perform multiple operations you must use multiple **WINDOW** commands.

Options:

Option	*4NT*	*TC*	*Description*
/POS=left,top,width,height	***x***	***x***	Set the window position and size on the desktop. The values are specified in pixels. ***Left*** and ***top*** refer to the position of the top left corner of the window relative to the top left corner (0,0) of the screen. The ***width*** and ***height*** values determine the window size. The window may be so sized and positioned that parts of it are beyond the physical area of the display. The **/** before the keyword is optional.
/TRANS=n		***x***	Set the transparency level Take Command window. ***n*** is an value from 0 (invisible) to 255 (opaque).
newtitle	***x***	***x***	Changes the window title. The title text must be enclosed in double quotes. (The quotes will not appear as part of the actual title as displayed.) Setting the title inside a batch file will only change the window title while the batch file is executing.
MAX	***x***	***x***	Expands the window to its maximum size.
MIN	***x***	***x***	Reduces the window to an icon.
RESTORE	***x***	***x***	Returns the window to its default size and location.
HIDE	***x***	***x***	Makes the window invisible. Use **RESTORE** to make the window visible.
TRAY	***x***	***x***	Moves the window to the taskbar tray.

/SIZE=rows,columns	***x***		Specify the ***4NT*** 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.
FS	***x***		Switches a ***4NT*** window to full-screen mode. Must use **WIN** (or **Alt-Enter**) to return to windowed mode.
WIN	***x***		Switches a full-screen ***4NT*** display to windowed mode.
TOPMOST		***x***	Keeps the ***TC*** window on top of all other windows until it closes, or **NOTOPMOST** is used.
NOTOPMOST		***x***	Allows other windows to overlay the ***TC*** window (this is the normal state for most windows).
TOP		***x***	Moves the ***TC*** window to the top of the window order, above all other non-**TOPMOST** windows.
BOTTOM		***x***	Moves the ***TC*** window to the bottom of the window order.

7.138 WMIQUERY

Purpose: Query the Windows Management Interface

Format: WMIQUERY [/A /B /C /H] namespace "query string" [index]

/A(ll instances) 394 /C(lasses) 394
/B(lank) 394 /H(eader) 395

namespace	The namespace to query
"query string"	WQL query string
index	Class instance

Usage:

You can use a single period **.** for ***namespace*** to default to **root\cimv2**.

For more details on what is available, see the WMI and WQL documentation on MSDN (msdn.microsoft.com), and download the "WMI Code Creator" from Microsoft and browse the available namespaces, classes, and properties.

For example, to query the ***name*** property from the ***Win32_Processor*** class:

```
wmiquery root\cimv2 "SELECT name FROM Win32_Processor"
```

To query available classes:

```
wmiquery /A root "select name from __namespace"
```

Options:

/A Display all class instances starting at "index".

/B Separate class instances with a blank line.

/C Display all the matching class names for the specified namespace. "***query string"*** is the filter to apply to the returned values; it can contain wildcards. For example:

```
wmiquery /c ."win32_q*"
```

/H Display a header for class instances.

7.139 Y

Purpose: Copy standard input to standard output, and then copy the specified file(s) to standard output.

Format: Y file ...

file The file or list of files to send to standard output.

See also: TEE 376, piping 39 and redirection 36.

Usage:

The Y command copies input from standard input (usually the keyboard) to standard output (usually the screen). Once the input ends, the named files are appended to standard output.

For example, to get text from standard input, append the files *MEMO1* and *MEMO2* to it, and send the output to *MEMOS*:

```
y memo1 memo2 > memos
```

The Y command is most useful if you want to add redirected data to the beginning of a file instead of appending it to the end. For example, this command copies the output of DIR, followed by the contents of the file DIREND, to the file DIRALL:

```
dir | y direnд > dirall
```

If you are typing at the keyboard to produce input text for Y, you must enter a **Ctrl-Z** to terminate the input.

When using Y with a pipe you must take into account that the programs on the two ends of the pipe run simultaneously, not sequentially.

See Piping 39 for more information on pipes.

8 Variables & Functions

The environment is a collection of information about your system that every program receives. Each entry in the environment consists of a variable name and a string value.

Usage

You can automatically substitute the text for the variable name in any command. To create the substitution, include a percent sign **%** and the variable name on the command line or in an alias or batch file, e.g., **%comspec**. If the name of the variable whose value you want to use is an expression, you can enclose the expression in brackets, e.g., **%[%n]**.

You can create, alter, view, and delete environment variables with the SET 351, ESET 258, and UNSET 388 commands, and in ***TC***, with the Environment popup window 81 available from the Utilities menu.

A few environment variables have special meanings for the command processor (they are listed in System Variables 398).

The command processor also supports two special types of variables:

- Internal variables 402 are similar to environment variables, but are interpreted internally by the command processor, 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 424 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 the command processor, you can use the FUNCTION 275 command to create your own. These latter ones are referred to as user defined functions or UDFs.

Note: The command processor inherits its initial environment from the process which started it. That process might be Explorer or another existing Windows process which launched the current command processor session. Note that if the starting process's environment is changed (through registry modifications, for example) while the command processor is already running, those changes will not be automatically reflected in the command processor's current environment. See the SET 351 command for details.

You use the SET 351 command to create a new environment variable. SET 351 can also modify or delete a single environment variable, or display the value of one or more environment variables. ESET 258 allows you to edit an environment variable. UNSET 388 deletes environment variables. For example, you can create a variable named **BACKUP** like this:

```
set BACKUP=*.bak;*.bk!;*.bk
```

If you then type:

```
del %BACKUP
```

it is equivalent to having type the command:

```
del *.bak;*.bk!;*.bk
```

The environment variable names you use this way may contain any alphabetic or numeric characters, the underscore character _, and the dollar sign **$**. You can force acceptance of other characters by including the full variable name in square brackets, like this: **%[AB##2]**. You can also indirectly reference environment variables using square brackets. For example **%[%var1]** means "the contents of the variable whose name is stored in ***VAR1***". A variable referenced with this technique cannot contain more than 8,191 characters.

In addition, the command processor uses the environment to keep track of the default directory on each drive. Windows only tracks the default directory of the current drive; the command processor 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 351 command.

In ***4NT*** and ***TC***, the size of the environment is set automatically and expanded as necessary. You do not need to specify the size as you did with 4DOS or COMMAND.COM.

The trailing percent sign that was traditionally required for environment variable names is not usually

required by ***4NT*** and ***TC***, which accept any character that cannot be part of a variable name as the terminator. However, the trailing percent can be used to maintain compatibility.

The trailing percent sign is needed if you want to append variable values. The following examples show the possible interactions between variables and literal strings. First, create two environment variables called ONE and TWO this way:

```
set ONE=abcd
set TWO=efgh
```

Now the following combinations produce the output text shown:

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 166 for details.

Environment variables may contain alias names. The command processor will substitute the variable value for the name, then check for any alias name which may have been included within the value. For example, the following commands would generate a 2-column directory of the .TXT files:

```
alias d2 dir /2
set cmd=d2
%cmd *.txt
```

For compatibility with some peculiar syntax introduced in recent **CMD.EXE** versions, ***4NT*** and ***TC*** now support:

`%var:string1=string2%`	Substitutes the second string for all instances of the first string in the variable.
`%var:~x[,y]%`	Returns the substring starting at the xth character position (base 0) and continuing for y characters. If y is not specified, returns the remainder of the string. If x is negative, starts from the end of the string.

For string manipulations, we suggest you rely instead on the much more flexible Variable Functions 433.

8.1 System Variables

The variables below have special meaning for the ***4NT*** and ***TC***.

CDPATH 398	Directory navigation search list
CMDLINE 398	Command line after full expansion
COLORDIR 398	Directory colorization specification
COMSPEC 398	Command processor specification
FILECOMPLETION 399	File completion control variable
HISTORYEXCLUSION 399	List of commands excluded from the command history 49
PATH 399	Executable program location search list
PATHEXT 399	Ordered search list of extensions of executable programs
PROMPT 400	Command prompt format specification
RECYCLEEXCLUDE 400	List of files excluded from the recycle bin
TEMP 400	Directory for temporary files
TITLEPROMPT 400	Command processor window title bar specification
TMP 400	Directory for temporary files
TREEEXCLUDE 400	List of directories excluded from JPSTREE.IDX

8.1.1 CDPATH variable

CDPATH specifies where to search for directories specified in CD 211, CDD 213, and PUSHD 331 commands and in automatic directory changes 17. See the CDPATH feature 13 for details.

This feature is maintained for backwards compatibility, but has largely been replaced by Extended Directory Searches 14.

8.1.2 CMDLINE variable

CMDLINE is set by the command processor to the fully expanded text of the currently executing command line just before invoking any external command (*.PIF*, *.COM*, *.EXE*, *.BTM*, *.BAT* or *.CMD),* unless the command line is prefaced with @ to prevent echoing, in which case **CMDLINE** will be removed.

8.1.3 COLORDIR variable

COLORDIR controls directory display colors used by DIR 233. See Color-Coded Directories 238 for a complete description of the format of this variable.

8.1.4 COMSPEC variable

Many programs expect the value of **COMSPEC** to contain the full path and name of the current character-mode command processor, e.g.

```
c:\program files\jpsoft\4nt.exe
```

4NT automatically sets COMSPEC to point to itself on startup (***TC*** does not). If you need to run a program from ***4NT*** which utilizes COMSPEC to locate a command processor to process commands or batch files that are not compatible with ***4NT***, you may set COMSPEC to the command processor your program expects before you start it. Since ***4NT*** and ***TC*** do not use COMSPEC, it is not

necessary to restore COMSPEC after completion of the program.

8.1.5 FILECOMPLETION variable

FILECOMPLETION specifies the files made available during filename completion for selected commands. See Customizing Filename Completion 60 for a complete description of the format.

8.1.6 HISTORYEXCLUDE variable

HistoryExclude specifies which commands should be excluded from the History List 49. The syntax is:

```
HistoryExclude=cmd1[;cmd2[;cmd3[;...]]]
```

For example, to exclude the DEL 225 and FREE 274 commands, the **Notepad** program and the user-defined alias **MYDIR**:

```
HistoryExclude=del;free;c:\windows\system32\notepad.exe;mydir
```

See also: HISTORY 285.

8.1.7 PATH variable

The **PATH** variable specifies the list of directories that the command processor will search for executable files that aren't in the current directory. **PATH** is used by some application programs to find their own files. See the PATH 319 command for a full description of this variable, which can also be changed or modified with SET 351 and ESET 258.

Note: We strongly recommend that you always leave at least your **WINDOWS** and **SYSTEM32** directories in the **PATH**. The directory where ***4NT*** or ***TC*** resides need not be included in **PATH**.

8.1.8 PATHEXT variable

PATHEXT is expected to contain a list of extensions (including a leading period **.**), separated by semicolons. For example, to replicate the default extension list used by the command processor:

```
set pathext=.pif;.com;.exe;.btm;.bat;.cmd;.rex;.rexx
```

If you use a command in a batch file or at the command prompt and all of the following are true:

- the directive PathExt 135=Yes
- the command is not an alias
- the command is not an internal command
- the command is not a filename with an explicit extension (thus neither an executable extension nor a Windows file association is available)

then the command processor will search each directory listed in PATH 399 in turn for a file with its name matching the command and its extension matching one of the extensions in PATHEXT. The 1st directory in PATH 399 is searched first, then the 2nd, looking in each for each of the extensions in PATHEXT in the order listed.

Enabling PATHEXT affects only the standard path search. It does not affect searches for files with explicit extensions, which must have either a standard extension (see list above), or one which is either an executable extension 33 or a Windows association.

Caution: If you set the directive PathExt 135 =Yes in the .INI file 91, and fail to set the PATHEXT variable, path searches without an explicit extension will fail as there will be no extensions for which to search! (Windows XP does define a default value for the PATHEXT variable.)

8.1.9 PROMPT variable

PROMPT defines the command line prompt. It can be set or changed with the PROMPT 329, SET 351 and ESET 258 commands. See the PROMPT 329 command for details.

See also:: TITLEPROMPT 400.

8.1.10 RECYCLEEXCLUDE variable

RECYCLEEXCLUDE specifies files to be excluded from the Recycle Bin.

The syntax is:

RecycleExclude=*file1[;file2...]*

file1, file2, ... : file specifications, may include wildcards

For example, to exclude ***.lib**, ***.obj**, and ***.bak** files:

```
RecycleExclude=*.lib;*.obj;*.bak
```

See also: DEL/ERASE 225 command and the RecycleBin 139 directive.

8.1.11 TEMP variable

TEMP specifies the directory where the command processor should store temporary files, unless the TMP 400 variable exists. Many other programs also use TEMP to locate where they should place their temporary files.

8.1.12 TITLEPROMPT variable

TITLEPROMPT can be used to specify the contents of the command processor's window title. Modifying its value changes the displayed title immediately. Unsetting it does NOT affect the title. It may contain the special escape-sequences acceptable in PROMPT 329, and all internal variables and functions can be used to generate it.

See also: ACTIVATE 186, PROMPT 329, TITLE 380 and WINDOW 392.

8.1.13 TMP variable

If **TMP** is defined, it specifies the directory where the command processor should store temporary files (overriding TEMP 400). Some other programs also use TMP to define where they should place their temporary files.

8.1.14 TREEEXCLUDE variable

TreeExclude specifies which drives and directories to ignore when updating the *JPSTREE.IDX* file. The syntax is:

```
TreeExclude=dir1[;dir2[;dir3[;...]]]
```

Any specified drive/directory and all of its subdirectories will be excluded from ***JPSTREE.IDX*** update. For example, to exclude everything in **c:\windows, d:\temp\temp2**, and everything on drive **g:**

```
TreeExclude=c:\windows;d:\temp\temp2;g:\
```

Setting **TreeExclude** to the base directory of the target of a directory tree copy can speed up the copying considerably.

See also:: Extended Directory Searches 14 and CDD 213.

8.2 CMD.EXE Compatibility Variables

The variables below are used by some Microsoft command processors, but are ignored by JP software command processors. To see their usage by Microsoft and the alternate methods to achieve the same purpose in our command processors, review:

COPYCMD 401	CMD.EXE default options for COPY 216 command
DIRCMD 401	CMD.EXE default options for DIR 233 command

The variables below are identical to the similarly named internal variables, and are for CMD.EXE compatibility:

DATE	See _DATE 415
TIME	See _TIME 423

8.2.1 COPYCMD variable

The **COPYCMD** variable is used by some versions of CMD.EXE to hold default options for the COPY 216 command. ***4NT*** and ***TC*** do not directly support this variable, i.e, its value has no affect on internal commands. In general, it is more efficient to define several aliases, each including a different combination of options. For example, if you want the COPY command to default to prompting you before overwriting an existing file, you could use this alias:

```
alias COPY=`*copy /r`
```

If you wish to use or create a COPYCMD variable for compatibility with CMD.EXE, you can define an alias to append the contents of that variable to the COPY command:

```
alias COPY=`*copy %copycmd`
```

Now each time the COPY alias is executed, the current value of COPYCMD will modify the execution of the COPY command.

8.2.2 DIRCMD variable

The **DIRCMD** variable is used by some versions of CMD.EXE to hold default options for the DIR 233 command. ***4NT*** and ***TC*** do not directly support this variable, i.e, its value has no affect on internal commands. In general, it is more efficient to define several aliases, each including a different combination of options. For example, if you want the DIR command to default to a 2-column display with a vertical sort and a pause at the end of each page, you could use this alias:

```
alias DIR=`*dir /2 /p /v`I
```

If you wish to use or create a DIRCMD variable for compatibility with CMD.EXE, you can define the alias to append the contents of that variable to the DIR command:

```
alias DIR=`*dir %dircmd`
```

Now each time the DIR alias is executed, the current value of DIRCMD will modify the execution of the DIR command.

8.3 Internal Variables

Internal variables are special variables built into the command processor to provide information about your system. They are not stored in the environment, but can be accessed as if they were environment variables in interactive commands, aliases, and batch files.

The values of these variables are stored internally in the command processor, and cannot be changed with the SET 351, UNSET 388, ESET 258 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
```

Some internal variables have constant values. These variables are included for compatibility, to make it easier to write batch files and aliases which work with all of our command processors.

Variables which return a file or directory name from a volume that supports long filenames return it in the same case as it is stored. Returned names are not quoted automatically, you must add the quotes yourself if they are required by the syntax in which you use them.

Some variables return values based on information provided by your operating system. These variables will only return correct information if the operating system provides it. For example, _BATTERY 412 will not return accurate results if your operating system and Advanced Power Management drivers do not provide correct information on battery status to the command processor.

For a list of internal variables organized by general categories of use, see Internal Variables by Category 406.

Examples

You can use internal variables in a wide variety of ways depending on your needs. Here are just a couple of examples. For a more comprehensive set of examples see the *EXAMPLES.BTM* file which came with your command processor.

Store the current date and time in a file, then save the output of a DIR command in the same file:

```
echo Directory as of %_date %_time > dirsave
dir >> dirsave
```

Use the IFF 287 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
```

8.3.1 Variables by Name

+ 410	Substitutes command separator
= 410	Substitutes command processor escape character
! 409	Last argument of the previous command
? 185	Exit code, last external program
4ver 410	Command processor version
_? 410	Exit code, last internal command

acstatus 411	AC line status
afswcell 411	OpenAFS workstation cell
alt 411	Alt key depressed: 1, else 0
ansi 411	ANSI X3.64 status

batch 411	Batch nesting level
batchline 411	Line number in current batch file.
batchname 411	Full path and filename of current batch file.
batchtype 411	Type of the current batch file
battery 412	Battery status
batterylife 412	Remaining battery life, seconds
batterypercent 412	Remaining battery life, %
bdebugger 412	Batch debugger active: 1, else 0
bg 412	Background color at cursor position
boot 412	Boot drive letter, without a colon
build 412	Build number

capslock 412	CapsLock on: 1, else 0
cdroms 412	List of the CD-ROM drives
childpid 413	Process ID of most recent child process
ci 413	Current text cursor shape in insert mode
cmdline 413	Current command line
cmdproc 413	Command processor name
cmdspec 413	Full pathname of command processor
co 413	Current text cursor shape in overstrike mode
codepage 413	Current code page number
column 413	Current cursor column
columns 413	Virtual screen width
country 413	Current country code
cpu 414	CPU type

cpuusage 414	CPU time usage (percent)
ctrl 414	Ctrl key depressed: 1, else 0
cwd 414	Current drive and directory
cwds 414	Current drive and directory with trailing \
cwp 414	Current directory
cwps 414	Current directory with trailing \

date 415	Current date
datetime 415	Current date and time, yyyyMMddhhmmss
day 415	Current day of the month
detachpid 415	Process ID of most recent detached process
disk 415	Current drive
dname 415	Name of the description file.
dos 415	Operating system type
dosver 415	Operating system version
dow 416	Current day of the week, English, short
dowf 416	Current day of the week, English, full
dowi 416	Current day of the week as an integer
doy 416	Current day of the year
drives 416	List of the existing drives
dst 416	Daylight savings time: 1, else 0
dvds 416	List of the DVD drives

echo 416	Echo turned on: 1, else 0
editmode 416	0 if in overstrike mode, 1 if in insert mode
errorlevel 424	Exit code, last external program
execstr 416	@EXECSTR return code
exit 416	Reason for exiting command processor
expansion 416	SETDOS /X value

fg 417	Foreground color at cursor position
ftperror 417	Last FTP error code

hdrives 417	List of the fixed drives
hlogfile 417	Current history log file name
host 417	Host name of local computer.
hour 417	Current hour
hwprofile 417	Windows hardware profile if defined

idleticks 418	Milliseconds since the last user input
idow 418	Current day of the week, local language, short
idowf 418	Current day of the week, local language, full
iftp 418	IFTP session active: 1, else 0
iftps 418	IFTPS session active: 1, else 0
imonth 418	Current month name, local language, short
imonthf 418	Current month name, local language, full
ininame 418	Full pathname of the current INI file
ip 418	IP address(es) of local computer.
isodate 418	Current date in ISO 9601 format

kbhit 418	Keyboard input character is waiting: 1, else 0

lalt 418	left Alt key depressed: 1, else 0
lastdisk 419	Last valid drive
lctrl 419	Left Ctrl key depressed: 1, else 0
logfile 419	Current log file name
lshift 419	Left Shift key depressed: 1, else 0

minute 419	Current minute
month 419	Current month of the year as integer
monthf 419	Current month of the year, English, full

numlock 419	NumLock on: 1, else 0

openafs 420	OpenAFS installed: 1, otherwise 0
osbuild 420	Windows build number

pid 420	Command processor process ID (numeric)
pipe 420	Current process is running in a pipe: 1, else 0
ppid 420	Process ID of parent process

ralt 420	Right Alt key depressed: 1, else 0
rctrl 420	Right Ctrl key depressed: 1, else 0
ready 420	List of accessible drives
registered 420	Registered user name
row 420	Current cursor row
rows 421	Screen height
rshift 421	Right Shift key depressed: 1, else 0

scrolllock 421	ScrollLock on: 1, else 0
second 421	Current second
selected 421	(TC) First line of highlighted text
shell 421	Shell level
shells 421	Shell level (old style)
shift 422	Shift key depressed: 1, else 0
shralias 422	SHRALIAS is loaded: 1, else 0
startpath 422	Startup directory of current shell.
startpid 422	Process ID of most recent STARTed process.
stdin 422	STDIN redirected: 1, else 0
stdout 422	STDOUT redirected: 1, else 0
stderr 422	STDERR redirected: 1, else 0
stzn 422	Name of time zone for standard time
stzo 422	Offset in minutes from UTC for standard time
syserr 422	Latest Windows error code

time 423	Current time
transient 423	Current process is a transient shell: 1, else 0
tzn 423	Name of current time zone
tzo 423	Offset in minutes from UTC for current time zone

unicode 423	Shell uses unicode for redirected output: 1, else 0

virtualpc 423	Running inside VirtualPC: 1; else 0
vmware 423	Running inside VMWare: 1; else 0

windir 423	Windows directory pathname
winfgwindow 423	Title of foreground window.
winname 423	Name of local computer
winsysdir 423	Windows system directory pathname
winticks 423	Milliseconds since Windows was started
wintitle 424	Current window title
winuser 423	Name of current user.
winver 424	Windows version number
wow64 424	Running inside WOW64: 1; else 0

xpixels 424	Physical screen horizontal size in pixels

year 424	Current year
ypixels 424	Physical screen vertical size in pixels

8.3.2 Variables by Category

- Command processor status 407
- Compatibility 409
- Dates and times 408
- Drives and directories 408
- Error codes 409
- Hardware status 406
- Operating system and software status 407
- Screen, color, and cursor 408

The list below gives a one-line description of all Internal Variables 402 and a cross reference which selects a separate help topic on that variable. Many variables are simple enough that the one-line description is probably sufficient, but in most cases you should check for any additional information in the cross referenced explanation if you are not already familiar with a variable. You can also obtain help on any function with a **HELP variablename** command at the prompt. See the HELP 284 command for details

Hardware status

acstatus 411	AC line status
alt 411	Alt key depressed
battery 412	Battery status
batterylife 412	Remaining battery life, seconds
batterypercent 412	Remaining battery life, %
capslock 412	CapsLock on: 1, otherwise 0
cpu 414	CPU type
cpuusage 414	CPU time usage (percent)
ctrl 414	Ctrl key depressed: 1, otherwise 0
kbhit 418	A keyboard input character is waiting: 1, otherwise 0
lalt 418	left Alt key depressed: 1, otherwise 0
lctrl 419	left Ctrl key depressed: 1, otherwise 0
lshift 419	left Shift key depressed: 1, otherwise 0

numlock 419	NumLock on:r 1, otherwise 0
ralt 420	right Alt key depressed: 1, otherwise 0
rctrl 420	right Ctrl key depressed: 1, otherwise 0
rshift 421	right Shift key depressed: 1, otherwise 0
scrolllock 421	ScrollLock on: 1, otherwise 0
shift 422	Shift key depressed: 1, otherwise 0

Operating system and software status

! 409	Last argument of previous command
ansi 411	ANSI X3.64 status
boot 412	Boot drive letter, without a colon
codepage 413	Current code page number
country 413	Current country code
dos 415	Operating system type
dosver 415	Operating system version
host 417	Host name of local computer.
hwprofile 417	Windows hardware profile if defined
idleticks 418	Milliseconds since last user input
ip 418	IP address(es) of local computer.
osbuild 420	Windows build number
virtualpc 423	Running inside VirtualPC: 1; else 0
vmware 423	Running inside VMWare: 1; else 0
windir 423	Windows directory pathname
winfgwindow 423	Title of foreground window.
winname 423	Name of local computer
winsysdir 423	Windows system directory pathname
winticks 423	Milliseconds since Windows was started
wintitle 424	Current window title
winuser 423	Name of current user.
winver 424	Windows version number
wow64 424	Running in Windows 64: 1; else 0

Command processor status

4ver 410	Command processor version
batch 411	Batch nesting level
batchline 411	Line number in current batch file.
batchname 411	Full path and filename of current batch file.
batchtype 411	Type of the current batch file
bdebugger 412	Batch debugger active: 1, otherwise 0
build 412	Build number
childpid 413	Process ID of most recent child process
cmdline 413	Current command line
cmdproc 413	Command processor name
cmdspec 413	Full pathname of command processor
detachpid 415	Process ID of most recent detached process
dname 415	Name of the description file.
echo 416	Echo status
editmode 416	Insert mode: 1; else 0
exit 416	Reason for command processor exit
expansion 416	Current expansion mode (SETDOS /X)
hlogfile 417	Current history log file name

iftp 418	IFTP session active: 1, otherwise 0
iftps 418	IFTPS session active: 1, otherwise 0
ininame 418	Full pathname of the current INI file
logfile 419	Current log file name
pid 420	the command processor process ID (numeric)
pipe 420	Current process is running in a pipe: 1, otherwise 0
ppid 420	Process ID of parent process
registered 420	Registered user name
shell 421	Shell level
shells 421	Shell level (old style)
shralias 422	SHRALIAS is loaded: 1, otherwise 0
startpath 422	Startup directory of current shell.
startpid 422	Process ID of most recent STARTed process.
stdin 422	STDIN redirected: 1, otherwise 0
stdout 422	STDOUT redirected: 1, otherwise 0
stderr 422	STDERR redirected: 1, otherwise 0
transient 423	Current process is a transient shell: 1, otherwise 0
unicode 423	4NT/TC uses unicode for redirected output: 1, otherwise 0

Screen, color, and cursor

bg 412	Background color at cursor position
ci 413	Current text cursor shape in insert mode
co 413	Current text cursor shape in overstrike mode
column 413	Current cursor column
columns 413	Virtual screen width
fg 417	Foreground color at cursor position
row 420	Current cursor row
rows 421	Screen height
selected 421	(TC) First line of highlighted text
xpixels 424	Physical screen horizontal size in pixels
ypixels 424	Physical screen vertical size in pixels

Drives and directories

afswcell 411	OpenAFS workstation cell
cdroms 412	List of CD-ROM drives
cwd 414	Current drive and directory
cwds 414	Current drive and directory with trailing \
cwp 414	Current directory
cwps 414	Current directory with trailing \
disk 415	Current drive
drives 416	List of all available drives
dvds 416	List of DVD drives
hdrives 417	List of hard (fixed) drives
lastdisk 419	Last valid drive
openafs 420	OpenAFS service installed: 1, otherwise 0
ready 420	List of ready (accessible) drives

Dates and times

date 415	Current date
datetime 415	Current date and time, yyyyMMddhhmmss
day 415	Current day of the month

dow 416	Current day of the week, English, short
dowf 416	Current day of the week, English, full
dowi 416	Current day of the week as an integer
doy 416	Current day of the year
dst 416	Daylight savings time: 1, else 0
hour 417	Current hour
idow 418	Current day of the week, local language, short
idowf 418	Current day of the week, local language, full
imonth 418	Current month name, local language, short
imonthf 418	Current month name, local language, full
isodate 418	Current date in ISO 9601 format
minute 419	Current minute
month 419	Current month of the year as integer
monthf 419	Current month of the year, English, full
second 421	Current second
stzn 422	Name of time zone for standard time
stzo 422	Offset in minutes from UTC for standard time
time 423	Current time
tzn 423	Name of current time zone
tzo 423	Offset in minutes from UTC for current time zone
year 424	Current year

Error codes

? 409	Exit code, last external program
? 410	Exit code, last internal command
errorlevel 424	Exit code, last external program
execstr 416	Last @EXECSTR return code
ftperror 417	Last FTP error code
syserr 422	Latest Windows error code

Compatibility

= 410	Substitutes command processor escape character
+ 410	Substitutes command separator

8.3.3 ! (Variable)

! returns the last argument of the previous command. The command is retrieved from the history list, so this will not work in a batch file -- it's intended for aliases and command line work.

8.3.4 ? variable

If an **external** command (i.e., a program) has an ***exit code***, its value is stored in the ? variable when the program terminates. Additionally. some **internal** commands, e.g., DIR 233 - to emulate Microsoft's **CMD.EXE** - also set this variable to the same value they set the variable _? 410, 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 _? 410 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 424.

See also: _? 410

8.3.5 _? variable

_? contains the exit code of the last internal command. You must use or save this value immediately, because it is set by every internal command, including the one used to save it.

Result codes:

0	command was successful
1	a usage error occurred
2	another command processor error or an operating system error occurred
3	the command was interrupted by **Ctrl-C** or **Ctrl-Break**

This variable can also be set in a subroutine by the RETURN 340 command.

Note that in imitation of CMD.EXE some internal commands, e.g., DIR, also set the variables ? 409 and ERRORLEVEL 424 to the same value they set this variable. However, you are strongly urged to use this variable.

See also: ? 409

8.3.6 = pseudovariable

= is the current EscapeChar 117. Use this pseudovariable, instead of the actual escape character, if you want your batch files and aliases to work in all Jpsoft command processors or in other users' environment regardless of how the escape character is defined.

8.3.7 + pseudovariable

+ is the current command separator 110. Use this pseudovariable, instead of the actual command separator, if you want your batch files and aliases to work in other users' environment regardless of how the command separator is defined.

WARNING: **%+** should always be surrounded by spaces.

For example, if the command separator is an ampersand [&] (the default in ***4NT*** / ***TC***) both of the commands below will display "Hello" on one line and "world" on the next. However, if the command separator has been changed the first command will display "Hello & echo world", while the second command will continue to work as intended.

```
echo Hello & echo world
echo Hello %+ echo world
```

8.3.8 _4VER

_4VER returns the current command processor version (for example, 8.00). The current decimal separator is used to separate the major and minor version numbers (see DecimalChar 113 for details).

See also: _BUILD 412.

8.3.9 _ACSTATUS

_ACSTATUS returns the AC line status.

value	*meaning*
0	Offline
1	Online
unknown	Unknown

8.3.10 _AFSWCELL

_AFSWCELL returns the OpenAFS workstation cell.

See http://www.openafs.org for more information on OpenAFS.

8.3.11 _ALT

_ALT returns the status of the two ***Alt*** keys:

value	*status of selected key*
1	at least one is depressed
0	neither is depressed

8.3.12 _ANSI

_ANSI returns **1** if internal support for ANSI Std. X3.64 40 is enabled, **0** if not.

SETDOS /A	**ANSI Directive**	**_ANSI**
0 (default)	Auto (default)	Result of test
1	Yes	1
2	No	0

8.3.13 _BATCH

_BATCH returns the current batch file nesting level. It is **0** if no batch file is currently being processed.

8.3.14 _BATCHLINE

_BATCHLINE returns the current line number in the current batch file. It is **-1** if no batch file is active.

8.3.15 _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.

8.3.16 _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

8.3.17 _BATTERY

_BATTERY returns the battery charge status:

value	*meaning*
1	High
2	Low
4	Critical
8	Charging
128	No battery
unknown	Unknown

8.3.18 _BATTERYLIFE

_BATTERYLIFE returns either the number of seconds of battery life remaining, or **unknown**.

8.3.19 _BATTERYPERCENT

_BATTERYPERCENT returns the percentage of battery charge remaining (**0...100**), or **unknown**.

8.3.20 _BDEBUGGER

_BDEBUGGER returns **1** if the batch debugger is actively debugging a file, or **0** if it is not.

8.3.21 _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 552 for details.

8.3.22 _BOOT

_BOOT returns the boot drive letter, without a colon.

8.3.23 _BUILD

_BUILD returns the internal ***4NT*** or ***TC*** build number.

See also: _4VER 410.

8.3.24 _CAPSLOCK

_CAPSLOCK returns the current state of the ***capslock*** key:

value	*toggled status*
1	**ON**
0	**OFF**

8.3.25 _CDROMS

_CDROMS returns a space-delimited list of the CD-ROM drives on the system.

8.3.26 _CHILDPID

_CHILDPID returns the process ID of the most recent child process.

8.3.27 _CI

_CI returns the insert mode cursor shape, as a percentage (**0** to **100**). See SETDOS /S 354 and CursorIns 112.

8.3.28 _CMDLINE

_CMDLINE returns the current command line. (This is most useful in key aliases.) If you specify it on the command line, it returns the contents of the command line with the %_cmdline name removed.

8.3.29 _CMDPROC

_CMDPROC returns the name of the current command processor (***4NT*** or ***TCMD32***). This variable is useful if you have batch files running in more than one environment, and need to take different actions depending on the underlying command processor.

See also: _4VER 410, _BUILD 412, _CMDSPEC 413, _DOS 415, _DOSVER 415, _WINVER 424.

8.3.30 _CMDSPEC

_CMDSPEC returns the full pathname of the command processor.

8.3.31 _CO

_CO returns the overstrike mode cursor shape, as a percentage (**0** to **100)**.

See also SETDOS /S 354 and CursorOver 112.

8.3.32 _CODEPAGE

_CODEPAGE returns the current Windows code page.

See also CHCP 215.

8.3.33 _COLUMN

_COLUMN is the current cursor column. The leftmost column is numbered **0**.

See also _COLUMNS 413, _ROW 420, and _ROWS 421.

8.3.34 _COLUMNS

_COLUMNS returns the current number of virtual screen columns (for example, **80**).

(TC) See Resizing the Take Command Window 84 for additional details on the virtual screen width.

See also _COLUMN 413, _ROW 420, and _ROWS 421.

8.3.35 _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.

8.3.36 _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.

To determine the type, revision, stepping level, and other such details for advanced processors, use the @WININFO 496 or @WMI 500 function.

8.3.37 _CPUUSAGE

_CPUUSAGE returns the current CPU usage, as a percent (**0** to **100**).

8.3.38 _CTRL

_CTRL returns the status of the two ***Ctrl*** keys:

value	*status of selected key*
1	at least one is depressed
0	neither is depressed

8.3.39 _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 414, _CWP 414, _CWPS 414, @CWD 444, and @CWDS 444.

8.3.40 _CWDS

_CWDS returns the current working directory in the format ***d:\pathname***.

See also _CWD 414, _CWP 414, _CWPS 414, @CWD 444, and @CWDS 444.

8.3.41 _CWP

_CWP returns the current working directory in the format ***\pathname*** (without the drive letter).

See also _CWD 414, _CWDS 414, _CWPS 414, @CWD 444, and @CWDS 444.

8.3.42 _CWPS

_CWPS returns the current working directory in the format **\pathname** (without the drive letter).

See also _CWD 414, _CWDS 414, _CWP 414, @CWD 444, and @CWDS 444.

8.3.43 _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 418.

8.3.44 _DATETIME

_DATETIME returns the current date and time in the format yyyyMMddhhmmss. The date part is the same as _isodate 418 without separators.

8.3.45 _DAY

_DAY returns the current day of the month (1 to 31).

8.3.46 _DETACHPID

_DETACHPID returns the process ID of the most recent process launched by the DETACH 232 command.

8.3.47 _DISK

_DISK returns the current disk drive letter, without a colon (for example, **C**).

If the current directory is a UNC, **%_disk** will return the sharename.

8.3.48 _DNAME

_DNAME returns the name of the file used to store file descriptions. It can be changed with the DescriptionName 115 directive in the .INI file 91, or the SETDOS /D 354 command.

8.3.49 _DOS

_DOS returns the operating system and command processor type. Each JP Software command processor returns a different value depending on the operating system, as follows:

Platform	*4NT*	*TC*
Windows 2000	WIN2K	WIN32
Windows XP	WINXP	WIN32
Windows 2003	WINXP	WIN32

This variable is useful if you have batch files running in more than one environment, and need to take different actions depending on the underlying operating environment or command processor. If you want the current command processor name, use _CMDPROC 413. To differentiate between different versions of Windows within ***4NT*** / ***TC***, use the _WINVER 424 variable; to differentiate between different command processors, use the _CMDPROC 413 variable.

8.3.50 _DOSVER

_DOSVER returns the current operating system version. The current decimal character is used to separate the major and minor version numbers (see DecimalChar 113).

8.3.51 _DOW

_DOW returns the first three characters of the name of the current day of the week (**Mon**, **Tue**, **Wed**, etc.).

8.3.52 _DOWF

_DOWF returns the full name of the day of the week for the current date (**Monday**, **Tuesday**, etc.).

8.3.53 _DOWI

_DOWI returns the current day of the week as an integer (**1** = Sunday, **2** = Monday, etc.).

8.3.54 _DOY

_DOY returns the current day of the year (1 to 366).

8.3.55 _DRIVES

_DRIVES returns a space-delimited list of the existing drives in the format:

A: C: D: E:

8.3.56 _DST

_DST returns 1 if daylight savings time is in effect, or 0 if it is not.

8.3.57 _DVDS

_DVDS returns a space-delimited list of the DVD drives on the system.

8.3.58 _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 253 command and the BatchEcho 107 directive). The value returned by the _ECHO variable reflects the state applicable at the time the variable is queried.

8.3.59 _EDITMODE

_EDITMODE returns 0 if the line editor is in overstrike mode, or 1 if it is in insert mode.

8.3.60 _EXECSTR

_EXECSTR returns the integer return code of the last @EXECSTR 454 function.

8.3.61 _EXIT

_EXIT returns the reason for exiting the command processor:

0 EXIT command
2 CLOSE_EVENT
4 LOGOFF_EVENT
6 SHUTDOWN_EVENT

8.3.62 _EXPANSION

_EXPANSION returns the current expansion mode (i.e., SETDOS /X 354). 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**.

8.3.63 _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 552 for details.

8.3.64 _FTPERROR

_FTPERROR returns the error code of the last error reported by FTP 42. 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)

8.3.65 _HDRIVES

_HDRIVES returns a space-delimited list of the hard (fixed) drives on the system.

8.3.66 _HLOGFILE

_HLOGFILE returns the name of the current history log file (or an empty string if LOG /H is OFF). See LOG 303 for information on history logging.

8.3.67 _HOST

_HOST returns the host name for the local computer.

8.3.68 _HOUR

_HOUR returns the current hour (0 - 23).

8.3.69 _HWPROFILE

_HWPROFILE returns the name of the current Windows hardware profile.

8.3.70 _IDLETICKS

_IDLETICKS returns the number of milliseconds since the last user input.

8.3.71 _IDOW

_IDOW returns the 3-character abbreviation for the day of the week for the current date, in the current locale language.

8.3.72 _IDOWF

_IDOWF returns the full name for the day of the week for the current date, in the current locale language.

8.3.73 _IFTP

_IFTP returns **1** if an IFTP 288 session is active, **0** if it is not.

8.3.74 _IFTPS

_IFTPS returns **1** if an SSL IFTP 288 session is active, **0** if it is not.

8.3.75 _IMONTH

_IMONTH returns the abbreviated name for the current month, in the current locale language.

8.3.76 _IMONTHF

_IMONTHF returns the full name for the current month, in the current locale language.

8.3.77 _ININAME

_ININAME returns the fully qualified pathname of the INI file used by the current shell.

8.3.78 _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.

8.3.79 _ISODATE

_ISODATE returns the current system date, in ISO 9601 format (**yyyy-mm-dd**).

See also _DATE 415 and _DATETIME 415.

8.3.80 _KBHIT

_KBHIT returns **1** if one or more keystrokes are waiting in the keyboard buffer, or **0** if the keyboard buffer is empty.

8.3.81 _LALT

_LALT returns the status of the left ***Alt*** key:

value	*key status*

1	depressed
0	not depressed

See also _ALT 411.

8.3.82 _LASTDISK

_LASTDISK returns the last valid drive letter (without a colon).

8.3.83 _LCTRL

_LCTRL returns the status of the Left Ctrl key:

value	*key status*
1	depressed
0	not depressed

See also _CTRL 414.

8.3.84 _LOGFILE

_LOGFILE returns the name of the current log file (or an empty string if LOG is OFF). See LOG 303 for information on logging.

8.3.85 _LSHIFT

_LSHIFT returns the status of the left shift key:

value	*key status*
1	depressed
0	not depressed

See also _SHIFT 422.

8.3.86 _MINUTE

_MINUTE returns the current minute (0 - 59).

8.3.87 _MONTH

_MONTH returns the current numeric month of the year (1 to 12).

8.3.88 _MONTHF

_MONTHF returns the full name of the current month (**January**, **February**, etc.).

8.3.89 _NUMLOCK

_NUMLOCK reports the current of the ***numlock*** key:

value	*toggled status*
1	ON
0	OFF

8.3.90 _OPENAFS

_OPENAFS returns **1** if the OpenAFS 46 service is active, **0** if it is not.

See http://www.openafs.org for more information on OpenAFS.

8.3.91 _OSBUILD

_OSBUILD returns the Windows build number.

8.3.92 _PID

_PID returns the current process ID number.

8.3.93 _PIPE

_PIPE returns **1** if the current process is running inside a pipe, and **0** otherwise.

8.3.94 _PPID

_PPID returns the process ID number of the parent process.

8.3.95 _RALT

_RALT returns the status of the right Alt key:

value	*key status*
1	depressed
0	not depressed

See also _ALT 411.

8.3.96 _RCTRL

_RCTRL returns the status of the right Ctrl key:

value	*key status*
1	depressed
0	not depressed

See also _CTRL 414.

8.3.97 _READY

_READY returns a space-delimited list of the currently ready (accessible) drives in the format :

C: D: E:

8.3.98 _REGISTERED

_REGISTERED returns the registered name of the user or an empty string if ***4NT*** / ***Take Command*** isn't registered.

8.3.99 _ROW

_ROW returns the current cursor row (for example, **0** for the top of the window).

8.3.100 _ROWS

_ROWS returns the current number of screen rows in the ***4NT*** / ***TC*** window (for example, **25**).

8.3.101 _RSHIFT

_RSHIFT returns the status of the right Shift key:

value	*key status*
1	depressed
0	not depressed

See also _SHIFT 422.

8.3.102 _RUBYTYPE

_RUBYTYPE returns the type of the Ruby VALUE returned by the last @RUBY 488 call.

8.3.103 _RUBYVALUE

_RUBYVALUE returns the Ruby VALUE returned by the last @RUBY 488 call.

8.3.104 _SCROLLLOCK

_SCROLLLOCK reports the current state of ***scrolllock*** mode, which can be toggled using the ***scrolllock*** key:

value	*toggled status*
1	ON
0	OFF

8.3.105 _SECOND

_SECOND is the current second (0 - 59).

8.3.106 _SELECTED

_SELECTED ***(TC)*** returns the first line of text highlighted in the ***TC*** window. If no text has been highlighted, SELECTED returns an empty string.

8.3.107 _SHELL

_SHELL is the current shell instance identifier, one for each command processor. **_SHELL** will return 0 for a primary shell, or 1 (or higher) for a shell started by another ***4NT*** or ***TC*** process (either directly or via a pipe).

See Primary and Secondary Shells 535 for an explanation of those terms.

8.3.108 _SHELLS

_SHELLS is the current shell identifier, one for each command processor. _SHELLS duplicates the behavior of _SHELL in older versions (7.01) and earlier of ***4NT*** and ***TC***.

See Primary and Secondary Shells 535 for an explanation of those terms.

8.3.109 _SHIFT

_SHIFT is the status the two ***Shift*** keys:

value	*status of selected key*
1	at least one is depressed
0	neither is depressed

8.3.110 _SHRALIAS

_SHRALIAS returns **1** if SHRALIAS 361 is loaded, **0** if it is not.

8.3.111 _STARTPATH

_STARTPATH returns the startup directory for the current shell (not necessarily the same as the location of the executable!)

8.3.112 _STARTPID

_STARTPID returns the process ID of the most recent process launched by the START 364 command.

8.3.113 _STDIN

_STDIN returns **1** if STDIN points to the console, or **0** if it has been redirected.

8.3.114 _STDOUT

_STDOUT returns **1** if STDOUT points to the console, or **0** if it has been redirected.

8.3.115 _STDERR

_STDERR returns **1** if STDERR points to the console, or **0** if it has been redirected.

8.3.116 _STZN

_STZN returns the name of standard time in the current time zone.

See also _STZO 422, _TZN 423, and _TZO 423.

8.3.117 _STZO

_STZO returns the offset in minutes from UTC for standard time in the current time zone.

See also _STZN 422, _TZN 423, and _TZO 423.

8.3.118 _SYSERR

_SYSERR returns the error code of the last operating system error. You will need a technical or programmer's manual to understand these error values.

See the Windows System Errors 536 table in the Reference section for examples.

8.3.119 _TIME

_TIME returns the current system time in the format **hh:mm:ss**. The separator character may vary depending upon your country information.

8.3.120 _TRANSIENT

_TRANSIENT returns **1** if the current shell is transient (started with a **/C**, see Command Line Options 4 for details), or **0** otherwise.

8.3.121 _TZN

_TZN returns the name of the current time zone.

See also _STZO 422, _STZO 422, and _TZO 423.

8.3.122 _TZO

_TZO returns the offset in minutes from UTC for the current time zone.

See also _STZN 422, _STZO 422, and _TZO 423.

8.3.123 _UNICODE

_UNICODE returns **1** if the shell is currently using Unicode for redirected output, **0** otherwise.

8.3.124 _VIRTUALPC

_VIRTUALPC returns 1 if the command processor is running inside VirtualPC virtual machine.

8.3.125 _VMWARE

_VMWARE returns 1 if the command processor is running inside a VMWare virtual machine.

8.3.126 _WINDIR

_WINDIR returns the pathname of the Windows directory.

8.3.127 _WINFGWINDOW

_WINFGWINDOW returns the title of the foreground window.

8.3.128 _WINNAME

_WINNAME returns the computer name of the current system.

8.3.129 _WINSYSDIR

_WINSYSDIR returns the pathname of the Windows system directory.

8.3.130 _WINTICKS

_WINTICKS returns the number of milliseconds since Windows was started.

8.3.131 _WINUSER

_WINUSER returns the name of the user currently logged on.

8.3.132 _WINVER

_WINVER returns the current Windows version number. The current decimal character is used to separate the major and minor version numbers (see DecimalChar 113 for details).

8.3.133 _WINTITLE

_WINTITLE returns the title of the current window.

8.3.134 _WOW64

_WOW64 returns 1 if the command processor is running in the WOW64 environment (64-bit Windows).

8.3.135 _XPIXELS

_XPIXELS returns the physical screen horizontal size in pixels.

8.3.136 _YEAR

_YEAR returns the current year (1980 to 2099).

8.3.137 _YPIXELS

_YPIXELS returns the physical screen vertical size in pixels.

8.3.138 ERRORLEVEL

ERRORLEVEL is an alternate name (included for compatibility with CMD.EXE) for the _? 410 variable, and is the exit code of the last external command. Many programs return **0** to indicate success and a non-zero value to signal an error. However, not all programs return an exit code. If no explicit exit code is returned, the value of **ERRORLEVEL** is undefined.

WARNING: For compatibility with CMD.EXE, some internal commands, e.g., **DIR**, also set this variable to the same value as the variable _? 410, 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 _? 410 variable instead.

See also _? 410

8.4 Variable Functions

Variable functions are very similar to internal variables, but they take one or more parameters (which can be environment variables or even other variable functions).

Variable functions are useful at the command prompt as well as in aliases 160 and batch files 162 to check on available system resources, manipulate strings and numbers, and work with files and filenames.

Many functions return values based on information provided by Windows. Such functions will only return correct information if the operating system provides it. For example, @READY 484 will not return accurate results if Windows does not provide correct disk drive status information to the command processor.

The variable functions built into the command processor are listed in alphabetical order in subsequent topics. You can also obtain help from the command prompt on any function with a **HELP @functionname** command, or by pressing Ctrl-F1 120 when the cursor is on the function name. See

the HELP 284 command for details

Note: The FUNCTION 275 command can be used to create, edit, or display user-defined variable functions, and the UNFUNCTION 387 to delete them.

For a list of Variable Functions organized by general categories of use, see Variable Functions by Category 431.

Syntax

To have either a user-defined or a built-in variable function evaluated, its name must be preceded by a percent sign **%** (**%@EVAL**, **%@LEN**, etc.). All variable functions must have square brackets **[ ]** enclosing their parameter(s), if any. No space is allowed between the function name and the **[**. The combined parameters of a variable function may not exceed 8,191 characters.

Memory Size / Disk Space / File Size Units and Report Format

Some variable functions, such as @DISKFREE 446, accept an optional parameter ***scale code***. These functions return a size of a disk or of an entity on the disk as a multiple of the specified scale factor from the table below. Lower case letters denote a power of 1,000, upper case letters a power of 1,024.

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

You can include **commas** (or the thousands separator 147 character) 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 JP Software command processors are case sensitive.

Date Parameter Format

See the Date Formats 72 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 446 or @READY 484, 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 460, @FILEWRITE 463, @FILEWRITEB 463, @FILESEEK 461, @FILESEEKL 461, and @FILECLOSE 458 functions allow you to access files based on their file handle. These functions must be used only with file handles returned by @FILEOPEN 459, 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 28 in commands.

Examples

You can use variable functions in a wide variety of ways depending on your needs. Here are a couple of examples to give you an idea of what's possible. For a much more comprehensive set of examples, see the file **EXAMPLES.BTM**, which comes with the command processor.

The command below sets the prompt to show the amount of free memory (see PROMPT 329 for details on including variable functions in your prompt):

```
prompt (%%@dosmem[K]K) $p$g
```

Set up a simple command line calculator. The calculator is used with a command such as CALC 3 * (4 + 5):

```
alias calc `echo The answer is:  %@eval[%$]`
```

(assuming a ParameterChar 135 value of "$").

8.4.1 Functions by Name

@ABS 437	Absolute value of number
@AFSCELL 437	OpenAFS cell name

@AFSMOUNT 437	OpenAFS mount point
@AFSPATH 437	Path in OpenAFS: 1, otherwise 0
@AFSSYMLINK 437	OpenAFS symbolic link
@AFSVOLID 437	OpenAFS volume ID
@AFSVOLNAME 437	OpenAFS volume name
@AGEDATE 438	Converts an age 555 into date and time
@ALIAS 438	Value of an alias
@ALTNAME 438	Short name for the file.
@ASCII 438	Set of ASCII-s for characters in string
@ASSOC 439	File association
@ATTRIB 439	Test or return file attributes
@AVERAGE 440	Average of a list of numbers

@CAPI 440	Call a _cdecl function in a DLL
@CAPS 440	Capitalize first character of each word
@CDROM 441	CD-ROM drive: 1, otherwise 0
@CEILING 441	Smallest integer not less than a number
@CHAR 441	Character string, given a set of ASCII-s
@CLIP 442	Specified line from clipboard
@CLIPW 442	Write string to the clipboard
@COLOR 442	RGB value of a color
@COMMA 442	Insert commas into a number (thousands separators)
@COMPARE 443	Two files are identical: 1, otherwise 0
@CONSOLE 443	Identify console sessions
@CONVERT 443	Convert value from input base to output base
@COUNT 443	Number of times a character appears in a string
@CRC32 443	File CRC
@CWD 444	Current Working Directory of specified drive
@CWDS 444	Current Working Directory of specified drive, with trailing backslash

@DATE 444	Convert date to number of days
@DAY 444	Day of month for date
@DEC 445	Decrement a numeric value by 1
@DECIMAL 445	Decimal portion of a number
@DESCRIPT 445	File description
@DEVICE 446	Character device: 1, otherwise 0
@DIGITS 446	String is all digits: 1, otherwise 0
@DIRSTACK 446	Directory stack entry
@DISKFREE 446	Free disk space
@DISKTOTAL 447	Total disk space
@DISKUSED 447	Used disk space
@DOMAIN 448	Domain name of a computer
@DOW 448	Short name of day of week for date
@DOWF 448	Full name of day of week for date
@DOWI 448	Day of week number for date
@DOY 449	Day of year for date
@DRIVETYPE 449	Type of a drive
@DRIVETYPEEX 449	Type of a drive

@ENUMSERVERS 450	Identify server names on a network
@ENUMSHARES 450	Identify sharenames on a server
@ERRTEXT 450	Windows error description
@EVAL 451	Arithmetic calculations
@EXEC 454	Execute a command and return its exit code
@EXECSTR 454	Execute a command and return the first output line
@EXETYPE 455	Application type
@EXPAND 455	All names that match filename
@EXT 456	File extension

@FIELD 456	Extract a field from a string
@FIELDS 457	Count fields in a string
@FILEAGE 457	File age 555 (date and time)
@FILECLOSE 458	Close a file handle
@FILEDATE 458	File date
@FILENAME 458	File name and extension
@FILEOPEN 459	Open a file handle
@FILEREAD 460	Read next line from a file handle
@FILES 460	Number of files matching a wildcard
@FILESEEK 461	Move a file handle pointer to specified file position
@FILESEEKL 461	Move a file handle pointer to a specified line
@FILESIZE 462	Total size of files matching a wildcard
@FILETIME 462	File time
@FILEWRITE 463	Write next line to a file handle
@FILEWRITEB 463	Write data to a file handle
@FINDCLOSE 464	Closes the search handle.
@FINDFIRST 464	Find first matching file
@FINDNEXT 465	Find next matching file
@FLOOR 465	Largest integer not larger than a number
@FORMAT 465	Formats data string according to format string
@FORMATN 466	Format a numeric value
@FSTYPE 466	File system type (FAT, NTFS, CDFS, etc.)
@FTYPE 466	Open command string for file type
@FULL 466	Full file name with path
@FUNCTION 467	Definition of a function

@GETDIR 467	Prompt for a directory name.
@GETFILE 467	Prompt for a path and file name.
@GETFOLDER 468	Folder name from tree view.
@GROUP 468	User is member of group: 1, otherwise 0

@HISTORY 468	A line or word from the command history

@IDOW 468	Short local name of day of week for date
@IDOWF 469	Full local name of day of week for date
@IF 469	Evaluates a conditional expression
@INC 470	Increment a numeric value by 1
@INDEX 470	Offset of string2 within string1
@INIREAD 471	Return an entry from an .INI file
@INIWRITE 471	Write an entry in an .INI file

@INSERT 472	Inserts string1 into string2
@INODE 472	File Inode (in hex)
@INSTR 473	Extract a substring
@INT 473	Integer part of a number
@IPADDRESS 473	Returns the numeric IP for a host name
@IPNAME 474	Returns the host name for a numeric IP address
@ISALNUM 474	Test for alphanumeric characters
@ISALPHA 474	Test for alphabetic characters
@ISASCII 474	Test for ASCII characters
@ISCNTRL 475	Test for control characters
@ISDIGIT 475	Test for decimal digits
@ISPRINT 475	Test for printable characters
@ISPUNCT 475	Test for punctuation characters
@ISSPACE 476	Test for white space characters
@ISXDIGIT 476	Test for hexadecimal digits

@JUNCTION 476	Directory referenced by the junction

@LABEL 476	Volume label
@LCS 476	Longest common sequence in two strings
@LEFT 476	Left end of string
@LEN 477	Length of a string
@LFN 477	Long name for a short filename
@LINE 477	Specified line from a file
@LINES 478	Count of lines in a file
@LINKS 478	Number of NTFS hard links for the file
@LOWER 478	Convert string to lower case
@LTRIM 478	Left trim specified characters.

@MAKEAGE 479	Convert date and time to age 555
@MAKEDATE 479	Convert number of days to date
@MAKETIME 479	Convert number of seconds to time
@MAX 480	Largest integer in the list
@MD5 480	MD5 hash of a string or file
@MIN 480	Smallest integer in the list
@MONTH 480	Month for date

@NAME 481	File name without path or extension
@NUMERIC 481	Test if a string is numeric

@OPTION 482	Current directive value
@OWNER 483	Return file owner

@PATH 482	File path without name
@PERL 483	Evaluate a Perl expression
@PING 483	Response time from a host

@QUOTE 483	Double quote the argument if necessary

@RANDOM 483	Generate a random integer
@READSCR 484	Read characters from the screen

@READY 484	Drive ready: 1, otherwise 0
@REGCREATE 484	Create registry subkey
@REGDELKEY 484	Delete a registry key and its subkeys
@REGEX 485	Match a regular expression
@REGEXINDEX 485	Return the offset of a regular expression match
@REGEXIST 485	Test if a registry key exists
@REGEXSUB 485	Return nth matching regular expression group
@REGQUERY 485	Read value from registry
@REGSET 485	Write value to registry
@REGSETENV 486	Write value to registry and broadcast change.
@REMOTE 486	Remote (network) drive: 1, otherwise 0
@REMOVABLE 486	Removable drive: 1, otherwise 0
@REPEAT 486	Repeat a character
@REPLACE 487	Replace string1 with string2 in text
@REVERSE 487	Reverse a string
@REXX 487	Value of executing an expression by REXX
@RIGHT 487	Right end of string.
@RTRIM 488	Removes specified trailing characters.
@RUBY 488	Evaluate a Ruby expression

@SCRIPT 488	Evaluate expression in an active scripting engine.
@SEARCH 488	Path search
@SELECT 489	Menu selection
@SERIAL 489	Serial number of a disk
@SFN 489	Short name for a long filename
@SHA1 490	SHA1 checksum for the file
@SHA256 490	SHA2-256 checksum for the file
@SHA384 490	SHA2-384 checksum for the file
@SHA512 490	SHA2-512 checksum for the file
@SIMILAR 490	Compare two strings for similarity
@SNAPSHOT 490	Save a window or desktop as a BMP
@STRIP 491	Strips all characters in char from string
@SUBST 491	Substitute a string within another string
@SUBSTR 491	Extract a substring
@SUMMARY 490	Query or set the NTFS SummaryInformation stream

@TIME 492	Convert a time of day to number of seconds
@TIMER 492	Get split time from timer.
@TRIM 492	Remove blanks from a string
@TRUENAME 492	Find true name of a file
@TRUNCATE 492	Truncate file at current position

@UNC 493	UNC name of a file
@UNICODE 493	Numeric UNICODE value for a character
@UNIQUE 493	Create file with unique name
@UNQUOTE 494	Remove double quotes from a filename
@UNQUOTES 494	Remove leading and trailing double quotes
@UPPER 494	Convert string to upper case

@VERINFO 494	Executable file version information

@WATTRIB 494	Test or return file attributes
@WILD 495	Compares strings using wildcards
@WINAPI 495	Call a Windows API function
@WINCLASS 496	Title of first window with classname
@WINEXENAME 496	Executable name for window
@WININFO 496	Current system information
@WINMEMORY 496	Windows memory information
@WINMETRICS 497	Windows system metrics
@WINPOS 498	Window position
@WINSTATE 498	Current state of window
@WINSYSTEM 499	Set/get windows system parameters
@WMI 500	Query WMI
@WORD 501	Extract a word from a string
@WORDS 502	Count words in a string
@WORKGROUP 502	Workgroup name of a computer

@XMLPATH 502	Year for date

@YEAR 502	Year for date

8.4.2 Functions by Category

This list gives a one-line description of all built-in Variable Functions 424, and a cross reference which selects a separate help topic on that function where you will find the detailed syntax and description. You can also obtain help on any function with a **HELP @functionname** command at the prompt or by pressing Ctrl-F1 120 when the cursor is on the function name. See the HELP 284 command for details

- Dates and times 435
- Drives and devices 432
- File content 432
- File names 433
- File properties 433
- Input dialog boxes 435
- Network properties 435
- Numbers and arithmetic 434
- Strings and characters 433
- System status 431
- Utility 436

Note: many functions have functionality that covers several categories.

System status

@ASSOC 439	File association for the extension
@CLIP 442	Specified line from clipboard
@CLIPW 442	Write string to the clipboard
@CONSOLE 443	Identify console sessions
@ERRTEXT 450	Windows error description
@FTYPE 466	Open command string for the file type
@READSCR 484	Read characters from the screen
@REGCREATE 484	Create registry subkey
@REGDELKEY 484	Delete a registry key and its subkeys
@REGEXIST 485	Test if a registry key exists
@REGQUERY 485	Read value from registry
@REGSET 485	Write value to registry
@REGSETENV 486	Write value to registry and broadcast change.
@WINCLASS 496	Title of first window with classname

@WINEXENAME 496	Executable name for window
@WININFO 496	Current system information
@WINMEMORY 496	Windows memory information
@WINMETRICS 497	Windows system metrics
@WINPOS 498	Window position
@WINSTATE 498	Current state of window
@WINSYSTEM 499	Set/get windows system parameters

Drives and devices

@CDROM 441	CD-ROM drive: 1, otherwise 0
@CWD 444	Current Working Directory of specified drive
@CWDS 444	Current Working Directory of specified drive, with trailing backslash
@DEVICE 446	Character device: 1, otherwise 0
@DISKFREE 446	Free disk space
@DISKTOTAL 447	Total disk space
@DISKUSED 447	Used disk space
@DRIVETYPE 449	Type of drive (hard drive, CD-ROM, etc.)
@DRIVETYPEEX 449	Type of drive (hard drive, CD-ROM, etc.)
@FSTYPE 466	File system type (FAT, NTFS, CDFS, etc.)
@JUNCTION 476	Directory referenced by the junction
@LABEL 476	Volume label
@READY 484	Drive ready: 1, otherwise 0
@REMOTE 486	Remote (network) drive: 1, otherwise 0
@REMOVABLE 486	Removable drive: 1, otherwise 0
@SERIAL 489	Serial number of a disk

File content

@COMPARE 443	Compare two files
@CRC32 443	File CRC
@FILECLOSE 458	Close a file handle
@FILEOPEN 459	Open a file handle
@FILEREAD 460	Read next line from a file handle
@FILESEEK 461	Move a file handle pointer
@FILESEEKL 461	Move a file handle pointer to a specified line
@FILEWRITE 463	Write next line to a file handle
@FILEWRITEB 463	Write data to a file handle
@INIREAD 471	Return an entry from an .INI file
@INIWRITE 471	Write an entry in an .INI file
@INODE 472	Inode value for a file
@LINE 477	Specified line from a file
@LINES 478	Count lines in a file
@LINKS 478	Number of NTFS hard links for a file
@MD5 480	MD5 hash of a string or file
@SHA1 490	SHA1 checksum for a file
@SHA256 490	SHA2-256 checksum for a file
@SHA384 490	SHA2-384 checksum for a file
@SHA512 490	SHA2-512 checksum for a file
@SUMMARY 490	NTFS SummaryInformation stream for a file

@TRUNCATE 492	Truncate file at current position
@VERINFO 494	Executable file version information

File names

@ALTNAME 438	Short name for the file.
@EXPAND 455	All names that match filename
@EXT 456	File extension
@FILENAME 458	File name and extension
@FULL 466	Full file name with path
@LFN 477	Long name for a short filename
@NAME 481	File name without path or extension
@PATH 482	File path without name
@QUOTE 483	Double quote a filename
@SFN 489	Short name for a long filename
@SEARCH 488	Path search
@TRUENAME 492	True name of a file
@UNC 493	UNC name of a file
@UNIQUE 493	Create file with unique name
@UNQUOTE 494	Remove double quotes from a filename
@UNQUOTES 494	Remove leading and trailing double quotes

File properties

@ATTRIB 439	Test or return file attributes
@DESCRIPT 445	File description
@EXETYPE 455	Application type
@FILEAGE 457	File age 555 (date and time)
@FILEDATE 458	File date
@FILES 460	Number of files matching a wildcard
@FILESIZE 462	Total size of files matching a wildcard
@FILETIME 462	File time
@FINDCLOSE 464	Closes the search handle.
@FINDFIRST 464	Find first matching file
@FINDNEXT 465	Find next matching file
@INODE 472	Inode value for a file
@LINKS 478	Number of NTFS hard links for a file
@OWNER 483	File owner
@SEARCH 488	Path search
@SUMMARY 490	NTFS SummaryInformation stream for a file
@TRUENAME 492	True name for a file
@UNIQUE 493	Create file with unique name
@VERINFO 494	Executable file version information
@WATTRIB 494	Test or return file attributes

Strings and characters

@ASCII 438	List of ASCII-s for characters in string
@CAPS 440	Capitalize first character of each word
@CHAR 441	Character string, given a set of ASCII-s
@COUNT 443	Counts occurrences of a character in a string
@EXECSTR 454	Execute a command and return its first output line

@FIELD 456	Extract a field from a string
@FIELDS 457	Count fields in a string
@FORMAT 465	Formats data string according to format string
@INDEX 470	Offset of string2 within string1
@INSERT 472	Insert string1 into string2
@INSTR 473	Extract a substring
@ISALNUM 474	Test for alphanumeric characters
@ISALPHA 474	Test for alphabetic characters
@ISASCII 474	Test for ASCII characters
@ISCNTRL 475	Test for control characters
@ISDIGIT 475	Test for decimal digits
@ISPRINT 475	Test for printable characters
@ISPUNCT 475	Test for punctuation characters
@ISSPACE 476	Test for white space characters
@ISXDIGIT 476	Test for hexadecimal digits
@LCS 476	Longest common sequence in two strings
@LEFT 476	Left end of string
@LEN 477	Length of a string
@LOWER 478	Convert string to lower case
@LTRIM 478	Trims specified leading characters.
@MD5 480	MD5 hash of a string or file
@REGEX 485	Return a Regular Expression test
@REGEXINDEX 485	Return the offset of a regular expression match
@REGEXSUB 485	Return the nth matching group of a regular expression test
@REPEAT 486	Repeat a character
@REPLACE 487	Replace string1 with string2 in text
@REVERSE 487	Reverse a string
@RIGHT 487	Right end of string
@RTRIM 488	Trims specified trailing characters.
@SIMILAR 490	Test similarity between two strings
@STRIP 491	Strips all characters in char from string
@SUBST 491	Substitute a string within another string
@SUBSTR 491	Older version of @INSTR 473 to extract a substring
@TRIM 492	Remove blanks from a string
@UNICODE 493	List of UNICODEs for characters in string
@UPPER 494	Convert string to upper case
@WILD 495	Compares strings using wildcards
@WORD 501	Extract a word from a string
@WORDS 502	Count words in a string

Numbers and arithmetic

@ABS 437	Absolute value of n
@AVERAGE 440	Average of a list
@CEILING 441	Smallest integer not less than n
@COMMA 442	Insert commas (thousands separators) into a numeric string
@CONVERT 443	Convert value from input base to output base
@DEC 445	Decrement a numeric value by 1
@DECIMAL 445	Decimal fraction portion of a number
@DIGITS 446	Tests if string is all digits
@EVAL 451	Arithmetic calculations

@FORMATN 466	Format a numeric value
@FLOOR 465	Largest integer not larger than n
@INC 470	Increment a numeric value by 1
@INT 473	Integer part of a number
@MAX 480	Largest integer in the list
@MIN 480	Smallest integer in the list
@NUMERIC 481	Test if a string is numeric
@RANDOM 483	Generate a random integer

Dates and times

@AGEDATE 438	Converts an age 555 into date and time
@DAY 444	Day of month for date
@DATE 444	Convert date to number of days
@DOW 448	Short name of day of week for date
@DOWF 448	Full name of day of week
@DOWI 448	Day of week as integer
@DOY 449	Day of year for date
@IDOW 468	Short localized name of day of week for date
@IDOWF 469	Full localized name of day of week for date
@MAKEAGE 479	Convert date and time to age 555
@MAKEDATE 479	Convert number of days to date
@MAKETIME 479	Convert number of seconds to time
@MONTH 480	Month in specified date
@TIME 492	Convert time to number of seconds
@YEAR 502	Year for date

Input dialog boxes

@GETDIR 467	Prompt for a directory name.
@GETFILE 467	Prompt for a path and file name.
@GETFOLDER 468	Folder name from tree view.
@SELECT 489	Menu selection

Network properties

@AFSCELL 437	OpenAFS cell name for a path
@AFSMOUNT 437	OpenAFS mount point for a path
@AFSPATH 437	Path is in OpenAFS: 1, otherwise 0
@AFSSYMLINK 437	OpenAFS symbolic link for a path
@AFSVOLID 437	OpenAFS volume ID for a path
@AFSVOLNAME 437	OpenAFS volume name for a path
@DOMAIN 448	Domain name of a computer
@ENUMSERVERS 450	Identify server names on a network
@ENUMSHARES 450	Identify sharenames on a server
@GROUP 468	User is member of group: 1, otherwise 0
@IPADDRESS 473	The numeric IP for a host name
@IPNAME 474	The host name for a numeric IP
@PING 483	Response time from a host
@WORKGROUP 502	Workgroup name of a computer

Utility

@ALIAS 438	Value of an alias
@CAPI 440	Call a _cdecl function in a DLL
@CLIP 442	Specified line from clipboard
@CLIPW 442	Write string to the clipboard
@COLOR 442	RGB value of a color
@DIRSTACK 446	Display directory stack entry
@ERRTEXT 450	Windows error description
@EXEC 454	Execute a command, returns its exit code
@EXECSTR 454	Execute a command, returns its first output line
@FUNCTION 467	Definition of a function
@HISTORY 468	A line or word from the command history
@IF 469	Value dependent on a conditional expression
@OPTION 482	Current .directive value
@PERL 483	Evaluate a Perl expression
@READSCR 484	Read characters from the screen
@REXX 487	Evaluate a REXX 175 expression
@RUBY 488	Evaluate a Ruby expression
@SCRIPT 488	Evaluate expression in active scripting engine
@SELECT 489	Menu selection
@SNAPSHOT 490	Save a window or the desktop to a BMP
@TIMER 492	Get split time from timer.
@WINAPI 495	Call a Windows API function
@WMI 500	Query WMI
@XMLPATH 502	Return text of XML element

8.4.3 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 9601

Field Order

For codes **1...4** the field order is as shown above. For code **0** the field order will also be one of those shown above. The command processor determines which field is reported first by Windows in a short date, and selects the order from the table above with the same first field. All other aspects of the Windows short date format are ignored,

Field Width

Month and day are always 2 digits. Year is 2 digits for codes **1**, **2** and **3**, and 4 digits for code **4**. For code **0** the year is 4 digits if it is the first field returned, and 2 digits if it is the last one.

Field Separator

If year is returned as 4 digits, the hyphen **-** is returned, regardless of any Windows settings. If year is returned as 2 digits, the Windows date separator is returned.

Setting the Windows Date Formats

The details below apply to Windows XP, but all other versions of Win32 are very similar.

Start ➔ Settings ➔ Control Panel ➔ Regional and Language Options ➔ Customize ➔ Date display the desired menu. The two relevant fields are **Short Date Format** and **Date Separator**.

8.4.4 @ABS

@ABS[*n*] : Returns the absolute value of the number ***n.***

Examples:

```
echo %@abs[-1]
echo %@abs[123]
```

8.4.5 @AFSCELL

@AFSCELL[*path*] : Returns the OpenAFS cell name for the path.

See http://www.openafs.org for more information on OpenAFS.

8.4.6 @AFSMOUNT

@AFSMOUNT[*path*] : Returns the OpenAFS mount point for the pathname.

See http://www.openafs.org for more information on OpenAFS.

8.4.7 @AFSPATH

@AFSPATH[*path*] : Returns 1 if the path is in the OpenAFS file system.

See http://www.openafs.org for more information on OpenAFS.

8.4.8 @AFSSYMLINK

@AFSSYMLINK[*path*] : Returns the OpenAFS symbolic link for the path.

See http://www.openafs.org for more information on OpenAFS.

8.4.9 @AFSVOLID

@AFSVOLID[*path*] : Returns the OpenAFS volume ID for the path.

See http://www.openafs.org for more information on OpenAFS.

8.4.10 @AFSVOLNAME

@AFSVOLNAME[*path*] : Returns the OpenAFS volume name for the path.

See http://www.openafs.org for more information on OpenAFS.

8.4.11 @AGEDATE

@**AGEDATE[***n[,d]***]** : Converts an age 555 ***n*** into a date and time pair, formatted according to the current country settings, or as explicitly specified by ***d*** (see Date Display Formats 436). The time is separated from the date by a comma, and is always in 24-hour format, displayed with 1 ms precision, as the examples show. The conversion does not take leap seconds into account.

Example:

```
for /l %n in (1,1,4) echo %n %@agedate[127551146920835000,%n]

1 03-12-05,15:24:52.083
2 12-03-05,15:24:52.083
3 05-03-12,15:24:52.083
4 2005-03-12,15:24:52.083
```

See also: Time Stamps 525, @FILEAGE 457 and @MAKEAGE 479.

8.4.12 @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 354 /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]
```

8.4.13 @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 570 of the full path.

Examples:

```
echo %@altname["Long Name.exe"]
echo %@altname["C:\Program Files\Microsoft Office"]
echo %@altname["%CommonProgramFiles"]
```

8.4.14 @ASCII

@**ASCII[***string***]** : Returns the space separated list of ASCII values of the characters in ***string***. You can use the EscapeChar 117 before a special character, e.g., a quote or greater than (>) sign, to include it in ***string***.

Note: The @UNICODE 493 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 540.

8.4.15 @ASSOC

@**ASSOC[**.*ext*] : Returns the file association for the specified extension.

Example:

```
echo %@assoc[.doc]
```

8.4.16 @ATTRIB

@**ATTRIB[***filename[,-rhsadecijopt[,p]]***]** : If you do not specify any attributes, @ATTRIB returns the attributes of the specified file in the format **RHSADECIJNOPT**, rather than **0** or **1**. If two or more parameters are specified, @ATTRIB returns a **1** if the specified file has the matching attribute(s); otherwise it returns a **0**.

The basic attributes for FAT volumes are:

N Normal (no attributes set)
R Read-only
A Archive
H Hidden
S System
D Directory

In addition, NTFS volumes allow display of the following extended attributes:

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

The extended attributes are displayed when @ATTRIB is invoked with a single parameter, but they are suppressed when used for file selection (two or more parameters). To select files based on the extended attributes, see @WATTRIB 494.

Attributes which are not set will be replaced with an underscore. For example, if *SECURE.DAT* has the read-only, hidden, and archive attributes set, **%@ATTRIB[SECURE.DAT]** would return RH_A_____. If the file does not exist, @ATTRIB returns an empty string.

The attributes (other than **N**) can be combined (for example %@ATTRIB[MYFILE,HS]). Normally @ATTRIB will only return **1** if all of the attributes match. However, if a final **,p** is included (partial match), then @ATTRIB will return **1** if any of the attributes match. For example, **%@ATTRIB[MYFILE,HS,p]** will return **1** if *MYFILE* has the hidden, system, or both attributes. Without **,p** the function will return **1** only if *MYFILE* has both attributes.

Filename must be in quotes if it contains white space or special characters.

See also: Attributes Switches 28.

Examples:

```
echo %@attrib["C:\Program Files\My Program\myfile.exe",rhs,p]
echo Attributes for myfile.exe: %@attrib[myfile.exe]
```

8.4.17 @AVERAGE

@**AVERAGE[**...**]** : Returns the average of a list of numbers. The average is returned as a double; you can adjust the decimal precision by running the result through @EVAL 451 (or @INT 473).

8.4.18 @CAPI

@**CAPI[**module,function[,integer | PING=n | PLONG=n | PDWORD=n | NULL | BUFFER | "string"**]** : Returns the result of calling a function with a _cdecl type in a DLL.

module - name of the DLL containing the function

function - function name (case sensitive)

integer - an integer value to pass to the function

PINT - a pointer to the integer ***n***

PLONG - a pointer to the long integer ***n***

PDWORD - a pointer to the DWORD ***n***

NULL - a null pointer (0)

BUFFER - @CAPI will pass an address for an internal buffer for the API to return a Unicode string value.

aBUFFER - @CAPI will pass an address for an internal buffer for the API to return an ASCII string value.

"**string**" - text argument (this must be enclosed in double quotes). If the argument is preceded by an 'a' (i.e., a"Argument") then it is converted from Unicode to ASCII before calling the API. (Some Windows APIs only accept ASCII arguments.)

@CAPI supports a maximum of 8 arguments. The return value is either a string value returned by the API (if BUFFER or aBUFFER is specified), or the integer value returned by the API. The function must be defined as _cdecl. If @CAPI can't find the specified function, it will append a "W" (for the Unicode version) to the function name and try again.

See also @WINAPI 495.

8.4.19 @CAPS

@**CAPS[***["xxx"],text***]** : Capitalizes the first letter of each word in the string (words that do not start with a letter remain unchanged). The optional first parameter, ***xxx***, specifies the separators that you wish

to use. The list must be enclosed in double quotes. If you want to use a double quote as a separator, prefix it with the Escape Character 69.

Examples:

```
echo %@caps[" ",i love 4nt and take command]
echo %@caps[",",peter,paul,mary]
echo %@caps[" ^"","sacrebleu!", he said]
```

8.4.20 @CDROM

@CDROM[*d:***]** : Returns **1** if the drive is an optical drive (CD-ROM, CD-RW, DVD, etc) or **0** otherwise. The drive letter must be followed by a colon.

Examples:

```
echo %@cdrom[C:]
echo %@cdrom[%_disk:]
```

8.4.21 @CEILING

@CEILING[*n***]** : Returns the value of the smallest integer that is not less than ***n.*** @CEILING will perform an implicit @EVAL 451 on its argument, so you can enter an arithmetic expression.

Examples:

```
echo %@ceiling[3.14]
echo %@ceiling[-3.14]
echo %@ceiling[0]
echo %@ceiling[123*37.36]
```

See also: @FLOOR 465.

8.4.22 @CHAR

@CHAR[*n***]** : Returns the character corresponding to a Unicode numeric value. If the parameter is a set of numeric values, CHAR returns a string. For example %@CHAR[65] returns A; %@CHAR[65 66 67] returns ABC.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

Note: Not all characters are printable. High ASCII characters (128-255) and Unicode characters may vary depending on the font used.

Examples:

```
echo %@char[65]
echo %@char[65 97 66 98 67 99]
```

8.4.23 @CLIP

@**CLIP[***n***]** : Returns line ***n*** from the Windows text clipboard. The first line is numbered 0. The string ****EOC**** is returned for all line numbers beyond the end of the clipboard.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

Examples:

```
echo %@clip[0]
if "%@clip[2]" eq "**EOC**" echo No more data in the clipboard
```

8.4.24 @CLIPW

@**CLIPW[***string***]** : Writes the ***string*** to the Windows text clipboard. Returns **0** if the operation was successful.

Examples:

```
if "%@clipw[save this line]" eq "0" echo Saved to the clipboard
```

8.4.25 @COLOR

@**COLOR[r,g,b]** : Displays the Windows color common dialog and returns the RGB value for the selected color as a string in the form **r,g,b** (e.g. **0,128,64**). To specify the initially selected color, use the **r** (red), **g** (green) and **b** (blue) parameters. If no parameters are provided, the initial selection will be black (**0,0,0**). The parameters are optional, but if one is used all three must be used.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

Examples:

```
@color[]
@color[155,0,0]
```

8.4.26 @COMMA

@**COMMA[***n***]** : Returns the number with commas (or the appropriate thousands separator 147 for your current country setting) inserted where appropriate.

Note: Some variable functions 424 can directly generate a numeric result with appropriate thousand separators if you add a ***c*** to their scale parameter.

Examples:

```
echo %@comma[12345678]
echo %@comma[0.12345678]
echo %@comma[%_xpixels]
```

See also: @CONVERT 443, @FORMAT 465, @FORMATN 466.

8.4.27 @COMPARE

@**COMPARE[***file1,file2***]** : Returns **1** if the two files are identical, or **0** if they differ. @COMPARE supports FTP filenames, but cannot compare two FTP files at the same time.

8.4.28 @CONSOLE

@**CONSOLE[***title***]** : Returns **1** if the specified window title belongs to a console window; **0** if it does not. The ***title*** may include wildcards 19.

8.4.29 @CONVERT

@**CONVERT[***input, output, value***]** : Returns a numeric string ***value*** converted from one number base (***input***) to another (***output***). Valid bases range from 2 to 36. The ***value*** can be between 0 and 2**64-1. No error is returned if ***value*** is outside that range.

Examples:

```
echo binary 1010101 is decimal %@convert[2,10,1010101]
echo decimal 20 is hex %@convert[10,16,20]
echo hexadecimal FF is octal %@convert[16,8,FF]
echo this year is %@convert[10,2,%_year] in binary
```

See also: @COMMA 442, @FORMAT 465, @FORMATN 466.

8.4.30 @COUNT

@**COUNT[***c,string***]** : Returns the number of times the character ***c*** appears in ***string***.

Examples:

```
echo %@count[a,Another function example]
```

8.4.31 @CRC32

@**CRC32[***filename***]** : Returns the CRC32 value (using the same algorithm as PKZIP or WINZIP) of the file specified by ***filename***, or **–1** if the file does not exist or cannot be opened.

See also: @SHA256 490, @SHA384 490, @SHA512 490, and @MD5 480..

Examples:

```
echo %@crc32["C:\My Files\Myprog.exe"]
echo %@crc32["%comspec"]
```

8.4.32 @CWD

@CWD[*d:***]** : Returns the current working directory of the specified disk drive in the format *d:\pathname*. If the current working directory is the root directory, the format is *d:*. The drive letter must be followed by a colon.

Examples:

```
echo %@cwd[C:]
echo %@cwd[%_disk:]
```

See also: @CWDS 444.

8.4.33 @CWDS

@CWDS[*d:***]** : Returns the current working directory of the specified disk drive in the format *d:\pathname*\. The drive letter must be followed by a colon.

Examples:

```
echo %@cwds[C:]
echo %@cwds[%_disk:]
```

See also: @CWD 444.

8.4.34 @DATE

@DATE[*date[,format]***]** : Returns the number of days since January 1, 1980 for the specified date. See date formats 72 for information on acceptable date formats. ***Date*** must be between 1980-01-01 and 2099-12-31 (inclusive).

@DATE accepts an optional second parameter specifying the date format:

0 default
1 USA (mm/dd/yy)
2 Europe (dd/mm/yy)
3 Japan (yy/mm/dd)
4 ISO (yyyy/mm/dd)

Examples:

```
echo %@date[01-01-1981]
echo %@date[%_date]
```

8.4.35 @DAY

@DAY[*date[,format]***]** : Returns the numeric day of the month for the specified date. See date formats 72 for information on acceptable date formats.

@DAY accepts an optional second parameter specifying the date format:

0 default
1 USA (mm/dd/yy)
2 Europe (dd/mm/yy)
3 Japan (yy/mm/dd)
4 ISO (yyyy/mm/dd)

Examples:

```
echo %@day[01-01-1980]
echo %@day[%_date]
```

8.4.36 @DEC

@DEC[*string***] :** Returns :

- **-1** if ***string*** is empty
- otherwise the same value as @EVAL 451[*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]
```

8.4.37 @DECIMAL

@DECIMAL[*number***]**: Returns the portion of ***number*** to the right of the decimal separator 113 as an integer numeric string. Trailing zeros are used to pad to the minimum width 117 specified for @EVAL 451. 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**

8.4.38 @DESCRIPT

@DESCRIPT[*filename***]**: Returns the file description for the specified filename (see DESCRIBE 230).

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"]
```

8.4.39 @DEVICE

@DEVICE[*name***]**: Returns **1** if the specified name is a character device (such as a serial port), or **0** if not.

Examples:

```
echo %@device[lpt1]
echo %@device[com1]
echo %@device[com5]
echo %@device[clip:]
echo %@device[clip]
```

8.4.40 @DIGITS

@DIGITS[*n***]**: Returns **1** if the string is composed of decimal digits only, otherwise it returns **0.** The DecimalChar 113, the ThousandsChar 147, and the sign characters (+ or -) are not digits, and if they are present in the string @DIGITS will return **0**.

Examples:

```
echo %@digits[12345]
echo %@digits[-12345]
echo %@digits[1.2345]
```

8.4.41 @DIRSTACK

@DIRSTACK[*n***]** : Returns the name of the ***n***th entry in the directory stack. The oldest is number 0. If no ***n*** parameter is specified, returns the total number of entries in the stack. The directory stack is set by calls to PUSHD 331 / POPD 326.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

See also: DIRS 245, POPD 326, PUSHD 331 and Directory Navigation 12.

Examples:

```
echo %@dirstack[0]
echo %@dirstack[2]
echo %@dirstack[]
```

8.4.42 @DISKFREE

@DISKFREE[*d:[,scale[c]]***]** : Returns the amount of free disk space on the specified drive. If you're specifying a drive, the drive letter must be followed by a colon. Optionally, you can specify a directory or UNC name, and @DISKFREE will display the free disk space on the drive referenced by that name (which may be different from the drive if the directory is a link to a directory on another drive).

The optional second parameter specifies the reporting scale (see Memory Size / Disk Space / File Size Units and Report Format 425). If the scale specification is suffixed with ***c*** the result will be formatted using the "thousands separator" (see ThousandsChar 147).

@DISKFREE supports OpenAFS 46 names.

See also: @DISKTOTAL 447 and @DISKUSED 447.

Examples:

```
echo %@diskfree[c:]
echo %@diskfree[%_disk:,Kc]
```

8.4.43 @DISKTOTAL

@DISKTOTAL[*d:[,scale[c]]***]** : Returns the total disk space on the specified drive. If you're specifying a drive, the drive letter must be followed by a colon. Optionally, you can specify a directory or UNC name, and @DISKTOTAL will display the total disk space on the drive referenced by that name (which may be different from the drive if the directory is a link to a directory on another drive).

The optional second parameter specifies the reporting scale (see Memory Size / Disk Space / File Size Units and Report Format 425). If the scale specification is suffixed with ***c*** the result will be formatted using the "thousands separator" (see ThousandsChar 147).

@DISKTOTAL supports OpenAFS 46 names.

See also: @DISKFREE 446 and @DISKUSED 447.

Examples:

```
echo %@disktotal[c:]
echo %@disktotal[%_disk:,Kc]
```

8.4.44 @DISKUSED

@DISKUSED[*d:[,scale[c]]***]** : Returns the amount of disk space in use on the specified drive. If you're specifying a drive, the drive letter must be followed by a colon. Optionally, you can specify a directory or UNC name, and @DISKUSED will display the disk space in use on the drive referenced by that name (which may be different from the drive if the directory is a link to a directory on another drive).

The optional second parameter specifies the reporting scale (see Memory Size / Disk Space / File Size Units and Report Format 425). If the scale specification is suffixed with ***c*** the result will be formatted using the "thousands separator" (see ThousandsChar 147).

@DISKUSED supports OpenAFS 46 names.

See also: @DISKFREE 446 and @DISKTOTAL 447.

Examples:

```
echo %@diskused[c:]
echo %@diskused[%_disk:,Kc]
```

8.4.45 @DOMAIN

@DOMAIN[*name***]** : Returns the domain of the computer specified by the DNS or NetBios ***name***. If ***name*** is not specified, returns the domain of the local computer.

8.4.46 @DOW

@DOW[*date[,format]***]** : Returns the first three characters of the English name of the day of the week for the specified date ("Mon", "Tue", "Wed", etc.). See date formats 72 for information on acceptable date formats.

@DOW accepts an optional second parameter specifying the date format:

0 default
1 USA (mm/dd/yy)
2 Europe (dd/mm/yy)
3 Japan (yy/mm/dd)
4 ISO (yyyy/mm/dd)

Examples:

```
echo %@dow[01-01-1980]
echo %@dow[%_date]
```

See also: @IDOW 468.

8.4.47 @DOWF

@DOWF[*date[,format]***]** : Returns the full English name of the day of the week for the specified date ("Monday", "Tuesday", etc.). See date formats 72 for information on acceptable parameter formats.

@DOWF accepts an optional second parameter specifying the date format:

0 default
1 USA (mm/dd/yy)
2 Europe (dd/mm/yy)
3 Japan (yy/mm/dd)
4 ISO (yyyy/mm/dd)

Examples:

```
echo %@dowf[01-01-1980]
echo %@dowf[%_date]
```

See also: @IDOWF 469.

8.4.48 @DOWI

@DOWI[*date[,format]***]** : Returns an integer representing the day of the week for the specified date (1 = Sunday, 2 = Monday, etc.). See date formats 72 for information on acceptable date formats.

@DOWI accepts an optional second parameter specifying the date format:

0 default
1 USA (mm/dd/yy)

2 Europe (dd/mm/yy)
3 Japan (yy/mm/dd)
4 ISO (yyyy/mm/dd)

Examples:

```
echo %@dowi[01-01-1980]
echo %@dowi[%_date]
```

8.4.49 @DOY

@**DOY[***date[,format]***]** : Returns the day of year (1 - 366) for the specified date. See date formats 72 for information on acceptable date formats.

@DOY accepts an optional second parameter specifying the date format:

0 default
1 USA (mm/dd/yy)
2 Europe (dd/mm/yy)
3 Japan (yy/mm/dd)
4 ISO (yyyy/mm/dd)

Examples:

```
echo %@doy[02-02-2005]
echo %@doy[%_date]
```

8.4.50 @DRIVETYPE

@**DRIVETYPE[***drive***]** : Return the type for the specified drive:

0 The drive type cannot be determined
1 The root path is invalid (no volume is mounted at the path)
2 Removable disk
3 Fixed disk
4 Remote (network) drive
5 CD-ROM
6 RAM disk

8.4.51 @DRIVETYPEEX

@**DRIVETYPEEX[***drive***]** : Return the type for the specified drive:

0 The drive type cannot be determined
1 The root path is invalid (no volume is mounted at the path)
2 Removable disk
3 Fixed disk
4 Remote (network) drive
5 CD-ROM
6 RAM disk
7 DVD
8 Tape

8.4.52 @ENUMSERVERS

@ENUMSERVERS[*n,server[,type]***]** : Enumerate the servers on the network. ***n*** is the entry number in the list of servers (the first one is **0**). ***server*** is the machine name(s) to match and it may contain wildcards 19. Returns a null string if there are fewer than ***n*** matching servers. This function can be repeatedly called, incrementing ***n*** each time to enumerate all available server names until it returns a null string.

If ***n*** is -1, @ENUMSERVERS returns the number of matching servers.

@ENUMSERVERS takes an optional third argument to return only servers of that type. The possible types are:

WORKSTATION - All workstations.
SQLSERVER - Any server running Microsoft SQL Server
DOMAIN - Primary domain controller
DOMAINBACKUP - Backup domain controller
DOMAIN_ENUM - Primary domain
LOCAL - Servers maintained by the browser
AFP - Apple File Protocol servers
TIME - Servers running the Timesource service
PRINTQ - Server sharing print queue
TERMINAL - Terminal Servers
CLUSTER - Server clusters in the domain
VSCLUSTER - Cluster virtual servers in the domain
MASTER - Server running the master browser service

WARNING! Windows may require a significant amount of time before returning data to this function when used on large networks. This is not anything over which ***4NT*** / ***TC*** has any control.

Examples:

```
echo %@enumservers[0,\\SERVER]
for %i in (0 1 2) echo %@enumservers[%i,*]
```

8.4.53 @ENUMSHARES

@ENUMSHARES[*n,\\server\shares***]** : Enumerate the share names for the specified server. ***n*** is the entry number in the list of shares (the first one is **0**). ***server*** is the server name, and ***shares*** is the sharename(s) to match. ***Shares*** may contain wildcards 19. Returns a null string If there are fewer than ***n*** matching shares. This function can be repeatedly called, incrementing ***n*** each time to enumerate all available shares until it returns a null string.

If the ***n*** is -1, @ENUMSHARES returns the number of matching sharenames.

Examples:

```
echo %@enumshares[0,\\SERVR\DRIVE_C]
for %i in (0 1 2) echo %@enumshares[%i,\\SERVER\DRIVE_*]
```

8.4.54 @ERRTEXT

@ERRTEXT[*n***]** : Returns the operating system error text for the specified code.

Examples:

```
echo %@errtext[2]
echo %@errtext[255]
echo %@errtext[%_syserr]
```

8.4.55 @EVAL

@EVAL[*expression[=displayformat]***]**: Evaluates a mathematical expression and returns its value in the format specified by ***displayformat*** or in the default format. Parameter Interpretation 451 below describes what ***expression*** may contain. Display precision and output format 453 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.

Parameter Interpretation

Expression may contain environment and internal variables and variable functions. After all variables and functions have been expanded, it must be composed only of numeric strings and names of functions in Trigonometric and transcendental functions 452, connected by Arithmetic operators 452 and optionally grouped with parentheses.

@EVAL permits you to simplify ***expression*** by dropping the **%** percent mark in front of the names of environment variables. You must include **%** for internal variables and variable functions. @EVAL also permits you to use characters which normally have special meaning for the command processor e.g., **& < > ^ |** without disabling their special meaning or quoting them.

Note: To ensure that ***expression*** is interpreted correctly, spaces should be placed on both sides of each operator, and parentheses used liberally. For example:

```
%@eval[(20 %% 3) + 4]
%@eval[12 and 65]
```

@EVAL also accepts numbers in the **e** exponent syntax; i.e. **1575e-2** = 15.75.

Number base

If a string starts with the characters **0x** it is interpreted as an integer in hexadecimal notation, and may be up to 16 digits long. Any other numeric string is considered to be a decimal number.

For example:

```
[c:\] echo %@eval[0x10 + 16]
```

```
32
```

You can specify hexadecimal output with the special syntax **@eval[...=H]**. For example:

```
echo %@eval[3*6=H]
```

will output 12 (hex). No leading 0x is included in the output. To convert between decimal and hexadecimal formats, see the @CONVERT 443 function.

Arithmetic operators

Every operator accepts both integer and non-integer parameters, except as noted below.

Operators accepting fractional parameters

+	(with one parameter) sign of numeric parameter (e.g. +3)
+	(with two parameters) addition
-	(with one parameter) negation of symbolic parameter (e.g., -%n) or sign of numeric parameter (*e.g.* –1, +3)
-	(with two parameters) subtraction
*****	multiplication
/	division
******	exponentiation

Operators requiring integer parameters

****	integer division (returns the integer part of the quotient)
MOD	modulo (returns the remainder when the first parameter is divided by the second)
%%	same as **MOD**
SHL	arithmetic left shift of the first parameter, truncated toward zero to an integer, by the number of bits specified by the second parameter
<<	same as **SHL**
SHR	arithmetic right shift of the first parameter, truncated toward zero to an integer, by the number of bits specified by the second parameter
>>	same as **SHR**

Operators which truncate parameters to integer

AND	bitwise **and** (returns 1 for each bit position where the corresponding bits in both parameters are 1)
&	same as **AND**
OR	bitwise **or** (returns 1 for each bit position where the corresponding bit in at least one parameter is 1)
|	same as **OR**
XOR	bitwise **exclusive or** (returns 1 for each bit position where the corresponding bits of the two parameters are different)
^	same as **XOR**
~	Unary **NOT**

Trigonometric and transcendental functions

Expression may include the trigonometric and transcendental functions below. The argument is interpreted as radians.

log(*x*) natural logarithm
log10(*x*) log 10
exp(*x*) exponential
sin(*x*) sine
asin(*x*) arcsine
sinh(*x*) hyperbolic sine
cos(*x*) cosine
acos(*x*) arccosine
cosh(*x*) hyperbolic cosine
tan(*x*) tangent
atan(*x*) arctangent
tanh(*x*) hyperbolic tangent

The special string **PI** is a shortcut for the value ***3.14159265358979323846***.

Order of precedence

1. variables,
2. expressions in matching parentheses,
3. functions listed in Trigonometric and transcendental functions 452 ,
4. exponentiation;
5. multiplication, division, and MOD;
6. addition and subtraction;
7. AND, OR, XOR, SHL, and SHR.

When multiple consecutive expressions of a single precedence level are used, evaluation is left to right.

For example, **3 + 4 * 2** will be interpreted as **3 + 8**, not as **7 * 2**. To change this order of evaluation, use parentheses to specify the order you want.

Precision of internal calculations

4NT and ***TC*** use 64-bit floating point arithmetic internally, i.e., each value (a "double") is represented by 64 bits: 1 for the sign, 11 for the exponent, 52 for the fraction part. This range is approximately **-1.7** to **1.7**. The effective precision of the result of **@EVAL** depends on which specific operations were performed on which values.

Note: Manipulating hexadecimal input and age 555 data without loss of significance may require greater precision than is available.

Display precision and output format

The maximum display precision is 20 digits to the left of the decimal separator and 10 digits to the right of the decimal separator. You can alter the default precision to the right of the decimal separator with the configuration dialogs, the EvalMax 117 and EvalMin 117 directives in the .INI file 91 and with the SETDOS 354 /F command. You can alter the decimal separator with the configuration dialogs, the DecimalChar 113 directive or the SETDOS 354 /G command. Any of these changes alter only the way a result is displayed, not how it is computed internally. If displayformat is not specified, display format is controlled by these default values

You can alter the display format for the current instance of @EVAL by specifying ***displayformat***.

Hexadecimal display format specification

If ***displayformat*** is the letter **H**, output will be hexadecimal. If ***displayformat*** is **X,** the output will be hexadecimal with a leading **0x**.

Explicit precision specification

If ***displayformat*** is ***i.a***, then:

- ***i*** must be a number which specifies the minimum decimal precision (the minimum number of decimal places displayed);
- ***a*** must be a number which sets the maximum decimal precision.
- the character separating ***i*** and ***a*** may be the comma if it is your decimal separator

You may specify either or both parameters ***i*** and ***a***.If ***i*** **>*a***, or if only ***i*** is specified, ***i*** is used as both the minimum and maximum precision, *e.g.* both **=2** and `=2.1` are equivalent to **=2.2.**

Examples

Expression	***Value***
`@eval[3 / 6=2.4]`	`0.50`
`@eval[3 / 6=4.4]`	`0.5000`
`@eval[3 / 7]`	`0.4285714286`
`@eval[3 / 7=.4]`	`0.4286`
`@eval[3 / 7=2.2]`	`0.42`
`@eval[3 / 7=2]`	`0.42`

See also: @DEC 445 and @INC 470.

8.4.56 @EXEC

@**EXEC[***command***]** : Execute ***command*** and return its numeric exit code.

Command can be an alias, internal command, external command, ***.****BTM*, ***.****BAT*, or ***.****CMD* file.

By default, @EXEC returns the result code from ***command*** (see the ? 409 and _? 410 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 454.

8.4.57 @EXECSTR

@**EXECSTR[***command***]** : Runs the specified ***command*** and returns the first line written to stdout 571 by ***command***.

@EXECSTR is useful for retrieving a result from an external utility — for example, if you have an external utility called **NETTIME.EXE** which retrieves the time of day from your network server and writes it to standard output, you could save it in an environment variable using a command like this:

```
set server_time=%@execstr[d:\path\nettime.exe]
```

If the same utility returned a result properly formatted for the TIME command, you could also use it to set the time on your system:

```
time %@execstr[d:\path\nettime.exe]
```

@EXECSTR can also be used with internal commands:

```
echo Newest file is: %@execstr[*dir /a:-d /h /o:-t /f]
```

@EXECSTR involves several extensive internal processing stages. You might be able to use more complex command sequences (pipes, command groups, etc.) as its parameter, but always *test* carefully first as the results may not always be what you expect. We recommend that you only use a single command (internal, external, batch file, etc.) parameter.

Note: Remember that only the first line of standard output is returned. Since many internal and external commands start their text output on the second line, @EXECSTR[] may not return any useful information from those commands.

See also: @EXEC 454.

8.4.58 @EXETYPE

@**EXETYPE[***filename***]**: Returns the application type for an executable file:

Code	*Application type*
0	Unknown
1	DOS app
2	PIF file
3	Win16
4	Win 3.x VxD
5	OS/2
6	Win32 GUI
7	Win32 console
8	Posix

Examples:

```
echo %@exetype["d:\path\myprog.exe"]
echo %@exetype["%comspec"]
```

8.4.59 @EXPAND

@**EXPAND[***[range...] filename[,[{+|-}]rhsadecijopt]***]** **:** Returns (in a single line), the names of all files and directories that are within the specified ***range[s]***, AND match ***filename***, AND have the specified attributes. ***Filename*** may contain wildcards 19 and include lists 30. Returns an empty string if no files match. If the file list is longer than the allowed command line length 71, it will be truncated without an error message. Each returned filename which contains white space or other special characters will be delimited by double quotes.

Filename must be in double quotes if it contains white space or special characters.

The ***range*** and attribute parameters, if included, define properties of the files that will be included in the result as specified in File Selection 18. Multiple ***range*** parameters may be included, but not more than one each of description range 28, size range 24, date range 24, and time range 26. ***Range*** parameters must precede ***filename***. Exclusion ranges 27 are not supported.

Examples:

```
echo %@expand[/[s2k,3k] *.txt]
```

all files with extension **txt** in the current directory with size at least 2000 bytes and at most 3000 bytes

```
echo %@expand[*,d]
```

all subdirectories

```
echo %@expand[/[d-365] %windir\w*.exe;w*.dll]
```

all files at most 365 days old in the Windows directory, with extension *EXE* or *DLL*, and name beginning with **W.**

8.4.60 @EXT

@**EXT[***filename***]** : Returns the extension from ***filename***, without a leading period. On volumes which support long file names, the extension can be up to 255 characters long. On FAT drives it can be up to 3 characters long. ***filename*** must be quoted if it contains white space or special characters.

On an LFN drive, the returned extension may contain white space or special characters. To avoid problems which could be caused by these characters, quote the returned extension before you pass it to other commands.

Examples:

```
echo %@ext[%@comspec]
echo %@ext["LFN Names may have.very long extensions"]
```

8.4.61 @FIELD

@**FIELD[***["sep_list",]n,string***]** : Returns the ***n***th field in ***string***. The first field is numbered **0**. If ***n*** is negative, fields are counted backwards from the end of ***string*.** You can specify the rightmost field by setting ***n*** to **-0**.

You can specify a range of fields to return with the syntax:

@FIELD[["sep_list",]start[-end | +range],string]

Specify an inclusive range with a **-**. For example:

%@FIELD[2-4,A B C D E F G] will return "C D E". (Note that you cannot use inclusive ranges when starting from the end.)

You can specify a relative range with a **+**. For example:

%@FIELD[2+1,A B C D E F G] will return "C D".

The default list of separators for @FIELD 456, @FIELDS 457, @WORD 501 and @WORDS 502 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 117, e.g., %=". Alphabetic characters in sep_list are case sensitive.

@FIELD 456 and @FIELDS 457 differ from @WORD 501 and @WORDS 502 in how multiple consecutive separators are counted. @WORD 501 and @WORDS 502 consider a sequence as a single separator, and ignore separators at either end of string. In contrast, @FIELD 456 and @FIELDS 457 count each occurrence of a separator individually, including those at either end of string.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits). To use hexadecimal form for a negative ***n***, remember to use 32-bit 2's complement arithmetic, e.g., **0xFFFFFFFF** for **-1**. There is no hexadecimal form to specify field **`-0`** (the rightmost field).

If ***string*** is double quoted, you must specify ***sep_list***.

See also: @WORD 501, @WORDS 502, @FIELDS 457.

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`

8.4.62 @FIELDS

@FIELDS[*["sep_list",]string***]** : Returns the number of fields in ***string***.

The default list of separators for @FIELD 456, @FIELDS 457, @WORD 501 and @WORDS 502 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 117, e.g., **%="**. Alphabetic characters in ***sep_list*** are case sensitive.

@FIELD 456 and @FIELDS 457 differ from @WORD 501 and @WORDS 502 in how multiple consecutive separators are counted. @WORD 501 and @WORDS 502 consider a sequence as a single separator, and ignore separators at either end of string. In contrast, @FIELD 456 and @FIELDS 457 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 501, @WORDS 502, @FIELD 456.

8.4.63 @FILEAGE

@FILEAGE[*filename[,a|c|w]***]** : Returns the date and time of the file as an age 555.

Filename must be in quotes if it contains white space or special characters. The optional second parameter selects which date field is returned for files on a VFAT 572 or NTFS 567 drive: ***a*** means the last access date, ***c*** means the creation date, and ***w*** means the last modification (write) date. The default is ***w***.

Examples:

```
echo %@fileage[d:\path\myfile.ext]
echo %@fileage["%comspec",c]
```

See also: Time Stamps 525, @AGEDATE 438 and @MAKEAGE 479.

8.4.64 @FILECLOSE

@FILECLOSE[*n***]** : Closes the file whose handle is ***n***. Returns **0** if the file was successfully closed, or **-1** if an error occurred.

This function should only be used with file handles returned by @FILEOPEN 459! If you use it with any other number you may damage other files opened by the command processor (or by the program which started the command processor).

See also the related handle-based functions:

@FILEOPEN 459	Open a file handle
@FILEREAD 460	Read next line from a file handle
@FILESEEK 461	Move a file handle pointer
@FILESEEKL 461	Move a file handle pointer to a specified line
@FILEWRITE 463	Write next line to a file handle
@FILEWRITEB 463	Write data to a file handle
@TRUNCATE 492	Truncate the file at the current position of the file handle pointer.

8.4.65 @FILEDATE

@FILEDATE[*filename[,a|c|w[,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 436). ***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.

Examples:

```
echo %@filedate["D:\my path\myfile.exe"]
echo %@filedate["comspec",c,4]
```

See Time Stamps 525, @FILETIME 462, @FILEAGE 457.

8.4.66 @FILENAME

@FILENAME[*filename***]** : Returns the name and extension of a file, without a path.

The ***filename*** must be in quotes if it contains white space or special characters. On an LFN drive, the returned filename may contain white space or other special characters. To avoid problems which could be caused by these characters, quote the returned name before you pass it to other commands.

Examples:

```
echo %@filename["D:\my path\myfile.exe"]
echo %@filename["comspec"]
```

8.4.67 @FILEOPEN

@FILEOPEN[*filename,r[ead]|w[rite]|a[ppend][,b|t]***]** : Opens the file in the specified mode and returns the file handle as an integer. The optional third parameter controls whether the file is opened in binary or text mode. Text mode (the default) should be used to read text using @FILEREAD 460 without a ***length***, and to write text using @FILEWRITE 463. Binary mode should be used to read binary data with @FILEREAD 460 with a ***length***, and to write binary data with @FILEWRITEB 463. Returns -1 if the file cannot be opened.

Filename must be in quotes if it contains white space or special characters.

To open a file for both reading and writing, open it in append mode, then use @FILESEEK 461 to position to the start of the file (or any other desired location) before performing additional operations.

@FILEOPEN can also open named pipes. The pipe name must begin with **\\.\pipe**. @FILEOPEN first tries to open an existing pipe; if that fails it tries to create a new pipe. Pipes are opened in blocking mode, duplex access, byte-read mode, and are inheritable. For more information on named pipes see your Windows documentation.

@FILEOPEN can open file streams on NTFS drives if the stream name is specified. See NTFS File Streams 526 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 exit the command processor, and restart it.

Examples:

```
set h=%@fileopen["d:\path\myfile.txt",write]
echo writing %@filewrite[%h,this is a test] bytes
echo closing handle #%h: %@fileclose[%h]
```

See also the related handle-based functions:

@FILECLOSE 458	Close a file handle
@FILEREAD 460	Read next line from a file handle
@FILESEEK 461	Move a file handle pointer
@FILESEEKL 461	Move a file handle pointer to a specified line
@FILEWRITE 463	Write next line to a file handle
@FILEWRITEB 463	Write data to a file handle
@TRUNCATE 492	Truncate the file at the current position of the file handle pointer.

8.4.68 @FILEREAD

@FILEREAD[*n[,length]***]** : Reads data from the file whose handle is ***n***. Returns the string ****EOF**** if you attempt to read past the end of the file. If ***length*** is not specified, **@FILEREAD** will read until the next CR or LF (end of line) character. If ***length*** is specified, @FILEREAD will read ***length*** bytes regardless of any end of line characters.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

This function should only be used with file handles returned by @FILEOPEN 459. If you use it with any other number you may damage other files opened by the command processor (or by the program which started the command processor).

Beware of characters with special meaning to the command processor, such as redirection and piping symbols, within the file. Use SETDOS 354 /X with appropriate codes as needed.

See also the related handle-based functions:

@FILECLOSE 458	Close a file handle
@FILEOPEN 459	Open a file handle
@FILESEEK 461	Move a file handle pointer
@FILESEEKL 461	Move a file handle pointer to a specified line
@FILEWRITE 463	Write next line to a file handle
@FILEWRITEB 463	Write data to a file handle
@TRUNCATE 492	Truncate the file at the current position of the file handle pointer.

8.4.69 @FILES

@FILES[*[range...] filename[,[{+|-}]rhsadecijopt]***]** : Returns the number of files within ***range*** that match ***filename*** and have the specified attributes. ***Filename*** may contain wildcards 19 and include lists 30. Returns 0 if no files match. To check files in multiple directories use @FILES once for each, and add the results with @EVAL 451.

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 18. Multiple ***range*** parameters may be included, but not more than one each of description range 28, size range 24, date range 24, and time range 26. ***Range*** parameters must precede ***filename***. Exclusion ranges 27 are not supported.

Examples:

```
echo %@files[/[s2k,3k] *.txt]
```
number of files with extension **txt** in the current directory with size at least 2000 bytes and at most 3000 bytes

```
echo %@files[*,d]
```
number of subdirectories

```
echo %@files[/[d-365] %windir\w*.exe;w*.dll]
```
number of files at most 365 days old in the Windows directory, with extension *EXE* or *DLL*, and name beginning with ***w***

8.4.70 @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 (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

This function should only be used with file handles returned by @FILEOPEN 459. If you use it with any other number you may damage other files opened by the command processor (or by the program which started the command processor).

Useful special cases

If you set ***offset*** to **0** :

- **@FILESEEK[*n*,0,0]** moves the file pointer to the beginning of file
- **@FILESEEK[*n*,0,1]** returns the current location of the file pointer without moving it.
- **@FILESEEK[*n*,0,2]** moves the file pointer to the end of file, and returns the current file size.

See also the related handle-based functions:

@FILECLOSE 458 Close a file handle
@FILEOPEN 459 Open a file handle
@FILEREAD 460 Read next line from a file handle
@FILESEEKL 461 Move a file handle pointer to a specified line
@FILEWRITE 463 Write next line to a file handle
@FILEWRITEB 463 Write data to a file handle
@TRUNCATE 492 Truncate the file at the current position of the file handle pointer.

8.4.71 @FILESEEKL

@FILESEEKL[*n,line[,1]***]** : Moves the file pointer to the specified ***line*** in the open file whose handle is ***n***. The first line in the file is numbered **0**. Returns the new position of the pointer, in bytes from the start of the file. The third parameter is optional, and determines the starting point for the seek. If not specified, or set to a value other than **1**, @FILESEEKL starts at the beginning of the file. If set to **1**, @FILESEEKL will start from the current position in the file.

If the function fails, the return value is **-1**.

@FILESEEKL must read each line of the file up to the target line in order to position the pointer, and can therefore cause significant delays if used in a loop or on a large file.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

This function should only be used with file handles returned by @FILEOPEN 459. If you use it with any other number you may damage other files opened by the command processor (or by the program which started the command processor).

See also the related handle-based functions:

@FILECLOSE 458	Close a file handle
@FILEOPEN 459	Open a file handle
@FILEREAD 460	Read next line from a file handle
@FILESEEK 461	Move a file handle pointer
@FILEWRITE 463	Write next line to a file handle
@FILEWRITEB 463	Write data to a file handle
@TRUNCATE 492	Truncate the file at the current position of the file handle pointer.

8.4.72 @FILESIZE

@FILESIZE[*filename[,[scale[c][,a]]***]** : Returns the size of a file, or **-1** if the file does not exist. If ***filename*** includes wildcards 19 or an include list 30, 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 425). Adding the letter ***c*** requests the result be formatted using the "thousands separator" (see ThousandsChar 147).

Examples:

```
echo %@filesize[d:\path\myfile.ext]
echo %@filesize["%comspec",bc]
echo %@filesize["%comspec",bc,a]
```

8.4.73 @FILETIME

@FILETIME[*filename[,[a|c|w][,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 525 for additional details).

Examples:

```
echo %@filetime["D:\my path\myfile.exe"]
echo %@filetime["comspec",c,s]
```

See also: @FILEDATE 458, @FILEAGE 457.

8.4.74 @FILEWRITE

@**FILEWRITE[***n,text***]**: Writes a line to the file whose handle is ***n***. Returns the number of bytes written, or **-1** if an error occurred.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

This function should only be used with file handles returned by @FILEOPEN 459. If you use it with any other number you may damage other files opened by the command processor (or by the program which started the command processor).

Beware of characters with special meaning to the command processor, such as redirection and piping symbols, within the file. Use SETDOS 354 /X with appropriate codes as needed.

See also the related handle-based functions:

@FILECLOSE 458	Close a file handle
@FILEOPEN 459	Open a file handle
@FILEREAD 460	Read next line from a file handle
@FILESEEK 461	Move a file handle pointer
@FILESEEKL 461	Move a file handle pointer to a specified line
@FILEWRITEB 463	Write data to a file handle
@TRUNCATE 492	Truncate the file at the current position of the file handle pointer.

8.4.75 @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.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

This function should only be used with file handles returned by @FILEOPEN! 459. If you use it with any other number you may damage other files opened by the command processor (or by the program which started the command processor).

Beware of characters with special meaning to the command processor, such as redirection and piping symbols, within the file. Use SETDOS 354 /X with appropriate codes as needed.

See also the related handle-based functions:

@FILECLOSE 458	Close a file handle
@FILEOPEN 459	Open a file handle
@FILEREAD 460	Read next line from a file handle

@FILESEEK 461	Move a file handle pointer
@FILESEEKL 461	Move a file handle pointer to a specified line
@FILEWRITE 463	Write next line to a file handle
@TRUNCATE 492	Truncate the file at the current position of the file handle pointer

8.4.76 @FINDCLOSE

@FINDCLOSE[*filename***]**: Signals the end of a @FINDFIRST 464 ... @FINDNEXT 465 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 464 ...@FINDNEXT 465 sequence is in effect, a non-zero value otherwise.

Examples:

```
echo %@findfirst[*.exe]
echo %@findclose[]
```

8.4.77 @FINDFIRST

@FINDFIRST[*[range...] filename[,[+|-]rhsadecijopt]***]** : Returns the name of the first file that matches ***filename***, which may include wildcards 19 and/or an include list 30, and which file has the properties specified in the optional range 22 and attribute 28 parameters.

Filename must be in quotes if it contains white space or special characters.

The ***range*** and attribute parameters, if included, define properties of the files that will be included in the search as specified in File Selection 18. Multiple ***range*** parameters may be included, but not more than one each of size range 24, date range 24, and time range 26. ***Range*** parameters must precede ***filename***. Each ***range*** parameter is of the form

```
/[a...]
```

where ***a*** is one of ***d***, ***s*** or ***t***, optionally followed by range parameters.

On an LFN drive, the returned filename may contain white space or other special characters. Unlike @EXPAND[] 455, 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 424 for additional details.

@FINDFIRST[] locates the *first* file matching the requirements. To find more matching files, you must use @FINDNEXT[] 465, and terminate the search with @FINDCLOSE[] 464.

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.

8.4.78 @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 464.

You do not need to include the ***filename*** parameter, because it must be the same as the one used in the previous @FINDFIRST call, unless you want to change the file attributes for @FINDNEXT. ***Filename***, if used, must be in quotes if it contains white space or special characters.

The attribute parameter, if included, defines the attributes of the files that will be included in the search as specified in Attribute Switches 28.

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[] 455, 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 424 for additional details.

@FINDFIRST[] locates the first file matching the requirements. To find more matching files, you must use @FINDNEXT[] 465, and terminate the search with @FINDCLOSE[] 464.

Examples:

```
echo %@findfirst[*]
echo %@findnext[]
echo %@findnext[*,d]
echo %@findclose[]
```

8.4.79 @FLOOR

@**FLOOR[***n***]**: Returns the largest integer that is not greater than ***n***. @FLOOR will perform an implicit @EVAL 451 on its argument, so you can enter an arithmetic expression.

Examples:

```
echo %@floor[3.14]
echo %@floor[-3.14]
echo %@floor[0]
echo %@floor[123]
```

See also: @CEILING 441.

8.4.80 @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.

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 442, @CONVERT 443, @FORMATN 466.

8.4.81 @FORMATN

@**FORMATN[***[-]width[.precision],value***]** : Formats a numeric value. ***Width*** is a nonnegative integer specifying the minimum number of characters printed. If ***Width*** has a leading **0**, the number will be left-padded with zeros. If the number of characters in the output value is less than the specified width, blanks are added to the left or the right of the values depending on whether the "**–"** flag (for left alignment) is specified, until the minimum width is reached. ***Precision*** specifies the number of digits after the decimal point. The ***value*** is rounded to the appropriate number of digits.

If you don't specify a precision, @FORMATN will default to 16 decimal places, and may not round the number appropriately. (For example, @FORMATN[3,3.4] will produce "3.3999999999999999".)

See also: @COMMA 442, @CONVERT 443, @FORMAT 465.

8.4.82 @FSTYPE

@**FSTYPE[**d:**]** : Returns the file system type for the specified drive. @FSTYPE returns **NTFS** for a drive that uses the Windows NTFS file system. It returns **FAT32** for FAT32 drives, and **FAT** for FAT12, FAT16, and VFAT drives.

Examples:

```
echo %@fstype[c:]
echo %@fstype[%_disk:]
```

8.4.83 @FTYPE

@**FTYPE**[*xxx*] : Returns the open command string for the specified file type.

Examples:

```
echo %@ftype[PerlScriptFile]
```

See also @ASSOC 439 and FTYPE 274.

8.4.84 @FULL

@**FULL[***filename***]** : Returns the full path and filename of a file. ***Filename*** must be in quotes if it contains white space or special characters. On an LFN drive, the returned filename may contain white space or other special characters. To avoid problems which could be caused by these characters, quote the returned name before you pass it to other commands. See the notes under Variable Functions 424 for additional details.

Note: The @FULL function makes no assumption about the existence of a file or directory. The ***filename*** parameter can be any string and the function will attempt to turn it into a fully qualified "volume + path + name" specification, whether that full reference exists or not.

Examples:

```
echo %@full[xyz.abc]
echo "%@full[.]"
echo "%@full["\Program Files"]"
```

8.4.85 @FUNCTION

@**FUNCTION[***name***]** : Returns the definition of the specified user-defined function 275 ***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 354. 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 275 command.

8.4.86 @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 424 for additional details.

@GETDIR accepts an optional second parameter to set the title of the dialog box.

Examples:

```
cdd %@getdir["C:\program Files"]
%@getdir[]\
```

Note: @GETDIR deals with directories. All directories are folders, but not all folders are directories. To select a symbolic folder, see @GETFOLDER 468.

8.4.87 @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 424 for additional details.

@GETFILE accepts an optional third parameter to set the title of the dialog box.

Examples:

```
echo %@getfile[*]
echo %@getfile["%windir",*.exe;*.com]
```

8.4.88 @GETFOLDER

@GETFOLDER[*startdir***]** : Returns a folder selected from a tree view of available symbolic folders. If you don't specify a start folder, @GETFOLDER starts at **My Computer** or the equivalent symbolic folder in your Windows configuration.

Examples:

```
echo %@getfolder[]
echo %@getfolder["Control Panel"]
```

Note: @GETFOLDER deals with folders. All directories are folders, but not all folders are directories. To select a directory, see @GETDIR 467.

8.4.89 @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.

8.4.90 @HISTORY

@HISTORY[*x[,y]***]** : Returns a line or word from the command history 49. (This function will prove most useful in keystroke aliases). ***x*** is the line to retrieve (current line = 0), and ***y*** is the specific word (first word = 0) desired within that line. If ***y*** is omitted, @HISTORY returns the entire line.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

8.4.91 @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 72 for information on date formats.

@IDOW accepts an optional second parameter specifying the date format:

0 default
1 USA (mm/dd/yy)
2 Europe (dd/mm/yy)

3 Japan (yy/mm/dd)
4 ISO (yyyy/mm/dd)

Examples:

```
echo %@idow[01-01-1980]
echo %@idow[%_date]
```

See also: @DOW 448.

8.4.92 @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 72 for information on date formats.

@IDOWF accepts an optional second parameter specifying the date format:

0 default
1 USA (mm/dd/yy)
2 Europe (dd/mm/yy)
3 Japan (yy/mm/dd)
4 ISO (yyyy/mm/dd)

Examples:

```
echo %@idowf[01-01-2007]
echo %@idowf[%_date]
```

See also: @DOWF 448.

8.4.93 @IF

@IF[*condition,string1,string2***]**: Evaluates ***condition*** according to the rules described in Conditional Expressions 53, 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
```

8.4.94 @INC

@INC[*string***]** returns

- **1** if ***string*** is empty
- otherwise the same value as @EVAL[451] [***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]
```

8.4.95 @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 EscapeChar[117] or its pseudovariable form %= in ***string2*** to escape[117] the comma
   `echo %@index["4NT, Take Command, 4DOS",%=,,2]`

- searching for a **double quote** :

1. use EscapeChar[117] or its pseudovariable form %= in ***string2*** to escape[117] 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[69] 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

8.4.96 @INIREAD

@**INIREAD[***file,section,entry***]**: Returns the value of the first matching ***entry*** from the specified ***file***, or an empty string if either ***file*** or the entry in ***file*** does not exist. If ***file*** contains more than one section named ***section***, only the first one is searched for ***entry***.

File, section, and ***entry*** must be in quotes if they contain white space or special characters.

File selection

Both the name and extension of ***file*** must be specified. This function does not apply a default extension. If ***file*** does not explicitly include a path, @INIREAD uses *%Windir*, the Windows installation directory. To use the current directory, you must explicitly specify it, e.g., using **.** as the path.

Example

```
%@iniread[c:\tcmd\tcmd32.ini,TakeCommand,history]
```

returns the size of the command history if it is specified in *TCMD32.INI*.

8.4.97 @INIWRITE

@**INIWRITE[***file,section,entry,string***]**: Creates, updates, or deletes an entry in the specified ***file***. If ***file*** does not exist, it will be created. @INIWRITE returns **0** for success or **-1** for failure.

File, section, and ***entry*** must be in quotes if they contain white space or special characters.

File selection

Both the name and extension of ***file*** must be specified. This function does not apply a default extension. If ***file*** does not explicitly include a path, @INIWRITE uses *%Windir*, the Windows installation directory. To use the current directory, you must explicitly specify it, e.g., using **.** as the path.

Action

If ***file*** does not exist, it will be created. If ***string*** is empty, ***file*** will be empty, otherwise a section line and a directive line will be created.

The remaining descriptions relate to the case when ***file*** exists.

If more than one match for ***section*** exists in ***file***, only the first one is searched for ***entry***. If more than one match exists for ***section*** and ***entry***, only the first match is one is affected. Searching starts at the beginning of the file, and stops on the first match.

If ***string*** is empty, the matching ***section*** and ***entry***, if any, is deleted. If ***string*** is not empty, and there is a matching ***section*** and ***entry***, it is modified. If ***string*** is not empty, and there is no matching ***section*** and ***entry***, it is created.

Examples

```
echo %@iniwrite[c:\tcmd\tcmd32.ini,TakeCommand,history,8192]
```

will set the size of the command history to 8,192 bytes.

```
echo %@iniwrite[c:\tcmd\tcmd32.ini,TakeCommand,history,]
```

will remove the ***history*** entry from the file.

8.4.98 @INODE

@**INODE**[*filename*] : Returns the inode (in hex) for the specified file.

When files are hard-linked to one another (see MKLNK 307), they share the same inode.

8.4.99 @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 EscapeChar 117.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits). To use hexadecimal form for a negative ***offset***, remember to use 32-bit 2's complement arithmetic.

Examples:

function	***value***
`%@insert[1,arm,wing]`	`warming`
`%@insert[8,very ,this is useful]`	`this is very useful`
`%@insert[255,%=, very!,this is useful]`	`this is useful, very!`
`%@insert[-9,very ,this is useful]`	`this very is useful`
`%@insert[0,abcde,xyz]`	`abcdexyz`

8.4.100 @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 491 is an older version of the same function.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits). To use hexadecimal form for a negative ***length***, remember to use 32-bit 2's complement arithmetic.

Examples:

function	***value***
`=%@instr[8,3,this is useful]=`	`=use=`
`=%@instr[8,,this is useful]=`	`=useful=`
`=%@instr[8,-4,this is useful]=`	`=is u=`
`=%@instr[8,,commas, they don't matter]=`	`=they don't matter=`

8.4.101 @INT

@**INT[***n***]**: Returns the integer part of the number ***n***. @INT will perform an implicit @EVAL 451 on its argument, so you can use an arithmetic expression for ***n.***

Examples:

```
echo %@int[1234]
echo %@int[1.234]
echo %@int[12.34]
```

8.4.102 @IPADDRESS

@**IPADDRESS[***hostname***]**: Returns the numeric IP address for the specified ***hostname***. If ***hostname*** is omitted, returns the current local IP address. The result is displayed in the common format **nnn.nnn.nnn.nnn**. An invalid or unknown ***hostname*** will return an error (see @ERRTEXT 450 to decipher the error number if necessary).

See also @IPNAME 474.

Example:

```
echo %@ipaddress[jpsoft.com]
echo %@ipaddress[]
```

8.4.103 @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 450 to decipher the error number if necessary).

See also:: @IPADDRESS 473.

Example:

```
echo %@ipname[192.220.109.228]
echo %@ipname[3235671524]
echo %@ipaddress[0]
```

8.4.104 @ISALNUM

@**ISALNUM[***string***]**: Returns **1** if ***string*** is entirely composed of alphabetic (a-z, A-Z) and/or numeric (0 - 9) characters; **0** otherwise.

See also:: @ISALPHA 474, @ISASCII 474, @ISCNTRL 475, @ISDIGIT 475, @ISPRINT 475, @ISPUNCT 475, @ISSPACE 476, @ISXDIGIT 476.

Example:

```
echo %@isalnum[123abc]
echo %@isalnum[123 abc]
echo %@isalnum[123.456]
```

8.4.105 @ISALPHA

@**ISALPHA[***string***]**: Returns **1** if ***string*** is entirely composed of alphabetic (a-z, A-Z) characters; **0** otherwise.

See also:: @ISALNUM 474, @ISASCII 474, @ISCNTRL 475, @ISDIGIT 475, @ISPRINT 475, @ISPUNCT 475, @ISSPACE 476, @ISXDIGIT 476.

Example:

```
echo %@isalpha[abc]
echo %@isalpha[ABC]
echo %@isalpha[A B C]
```

8.4.106 @ISASCII

@**ISASCII[***string***]**: Returns **1** if ***string*** is entirely composed of 7-bit ASCII characters (0x00 – 0x7F); **0** otherwise.

See also:: @ISALNUM 474, @ISALPHA 474, @ISCNTRL 475, @ISDIGIT 475, @ISPRINT 475, @ISPUNCT 475, @ISSPACE 476, @ISXDIGIT 476.

Example:

```
echo %@isascii[abc]
echo %@isascii[abc 123]
echo %@isascii["abc",123]
```

8.4.107 @ISCNTRL

@**ISCNTRL[***string***]**: Returns **1** if ***string*** is entirely composed of ASCII control characters (0x00 – 0x1F or 0x7F); **0** otherwise.

See also:: @ISALNUM 474, @ISALPHA 474, @ISASCII 474, @ISDIGIT 475, @ISPRINT 475, @ISPUNCT 475, @ISSPACE 476, @ISXDIGIT 476.

8.4.108 @ISDIGIT

@**ISDIGIT[***string***]** : Returns **1** if ***string*** is entirely composed of decimal digits (0– 9); **0** otherwise.

See also:: @ISALNUM 474, @ISALPHA 474, @ISASCII 474, @ISCNTRL 475, @ISPRINT 475, @ISPUNCT 475, @ISSPACE 476, @ISXDIGIT 476.

Example:

```
echo %@isdigit[0]
echo %@isdigit[123.456]
echo %@isdigit[-123]
```

8.4.109 @ISPRINT

@**ISPRINT[***string***]**: Returns **1** if ***string*** is entirely composed of printable characters; **0** otherwise.

See also:: @ISALNUM 474, @ISALPHA 474, @ISASCII 474, @ISCNTRL 475, @ISDIGIT 475, @ISPUNCT 475, @ISSPACE 476, @ISXDIGIT 476.

8.4.110 @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 474, @ISALPHA 474, @ISASCII 474, @ISCNTRL 475, @ISDIGIT 475, @ISPRINT 475, @ISSPACE 476, @ISXDIGIT 476.

Example:

```
echo %@ispunct[.]
echo %@ispunct[+]
echo %@ispunct[:-)]
```

8.4.111 @ISSPACE

@**ISSPACE[***string***]**: Returns **1** if ***string*** is entirely composed of white space characters (0x09 – 0x0D or 0x20); **0** otherwise.

See also:: @ISALNUM 474, @ISALPHA 474, @ISASCII 474, @ISCNTRL 475, @ISDIGIT 475, @ISPRINT 475, @ISPUNCT 475, @ISXDIGIT 476.

8.4.112 @ISXDIGIT

@**ISXDIGIT[***string***]**: Returns **1** if ***string*** is entirely composed of hexadecimal digits (0– 9,A-F, a-f); **0** otherwise.

See also:: @ISALNUM 474, @ISALPHA 474, @ISASCII 474, @ISCNTRL 475, @ISDIGIT 475, @ISPRINT 475, @ISPUNCT 475, @ISSPACE 476.

Example:

```
echo %@isxdigit[0]
echo %@isxdigit[7F]
echo %@isxdigit[0x7F]
```

8.4.113 @JUNCTION

@**JUNCTION[***dir***]** : Returns the directory referenced by the specified junction.

8.4.114 @LABEL

@**LABEL[***d:***]**: Returns the volume label of the specified disk drive. The drive letter must be followed by a colon.

Examples:

```
echo %@label[C:]
echo %@label[%_disk:]
```

See also: VOL 390.

8.4.115 @LCS

@**LCS[***string1,string2***]** : Returns a pointer to the longest common sequence in ***string1*** and ***string2***.

8.4.116 @LEFT

@**LEFT[***n,string***]** : If ***n*** is positive, it returns the leftmost ***n*** characters of ***string*.** If ***n*** is greater than the length of ***string***, it returns the entire ***string***. If ***n*** is negative, it returns ***string*** after dropping its rightmost ***n*** characters, unless ***-n*** is greater than the length of ***string***, in which case it returns an empty string.

Examples:

```
%@LEFT[2,jpsoft]   jp
%@LEFT[22,jpsoft]  jpsoft
%@LEFT[-2,jpsoft]  jpso
%@LEFT[-22,jpsoft] empty string
```

8.4.117 @LEN

@**LEN[***string***]** : Returns the length of ***string***.

Examples:

```
echo %@len[this is a test]
echo %@len[%comspec]
```

8.4.118 @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 424 for additional details.

Examples:

```
echo %@lfn[xyz.abc]
echo "%@lfn[c:\progra~1]"
```

8.4.119 @LINE

@**LINE[***filename,n***]**: Returns line ***n*** from the specified file. The first line in the file is numbered 0. ****EOF**** is returned for all line numbers beyond the end of the file.

The ***filename*** must be in quotes if it contains white space or special characters.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

@LINE works with files having lines of no more than 8,191 characters. Longer lines will not be counted accurately.

The @LINE function must read each line of the file to find the line you request, and will therefore cause significant delays if used in a long loop or on a large file. For a more effective method of processing each line of a file in sequence use the DO 246 command, or @FILEOPEN 459 and a sequence of @FILEREADs. 460

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.

Beware of characters with special meaning to the command processor, such as redirection and piping symbols, within the file. Use SETDOS /X with appropriate codes as needed.

8.4.120 @LINES

@LINES[*filename***]**: Returns the line number of the last line in the file, or "-1" if the file is empty. The first line in the file is numbered 0, so (for example) @LINES will return 0 for a file containing one line. To get the actual number of lines, use %@INC[%@LINES[filename]].

The ***filename*** must be in quotes if it contains white space or special characters.

@LINES works with files having lines of no more than 8,191 characters; longer lines will not be counted accurately. @LINES must read each line of the file in order to count it, and will therefore cause significant delays if used on a large file.

@LINES can count lines in file streams on NTFS drives if the stream name is specified. See NTFS File Streams for additional details on file streams.

8.4.121 @LINKS

@LINKS[*filename*] : Returns the number of hard links for the specified file (NTFS only).

See also MKLNK.

8.4.122 @LOWER

@LOWER[*string***]** : Returns the ***string*** converted to lower case.

Examples:

```
echo %@lower[ThiS iSS aTeSt]
echo %@lower[%path]
```

8.4.123 @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"
```

8.4.124 @MAKEAGE

@MAKEAGE[*date[,time[,format]]***]** : Converts ***date*** and ***time*** (if included) to an age 555, a single value in the same format as @FILEAGE 457.

@MAKEAGE accepts an optional third parameter specifying the date format:

0 default
1 USA (mm/dd/yy)
2 Europe (dd/mm/yy)
3 Japan (yy/mm/dd)
4 ISO (yyyy/mm/dd)

@MAKEAGE can be used to compare the time stamp of a file with a specific date and time, for example:

```
if %@fileage[myfile] lt %@makeage[1/1/85] echo OLD!
```

@AGEDATE 438 is the inverse of this function.

Examples:

```
echo %@makeage[%_date]
echo %@makeage[%_date,%_time]
```

See also: Time Stamps 525, @FILEAGE 457, @AGEDATE 438.

8.4.125 @MAKEDATE

@MAKEDATE[*n[,d]***]**: Returns a date, formatted according to the current country settings, or as explicitly specified by ***d*** (see Date Display Formats 436). ***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 444. The optional second parameter specifies the date format:

0 system default
1 USA (mm/dd/yy)
2 European (dd/mm/yy)
3 Japan (yy/mm/dd)
4 ISO 9601 (yyyy-mm-dd)

Examples:

```
echo %@makedate[7924]
echo %@makedate[7924,4]
```

8.4.126 @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 492.

Examples:

```
echo %@maketime[45240]
echo %@maketime[79244]
```

8.4.127 @MAX

@**MAX[***a,b,c,...***]**: Returns the largest in the list of parameters. All parameters must be integers in the range **-2147483647** `to` **2147483647** and must be separated either by whitespace or by commas.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

Example:

```
echo %@max[1,5,2,0,-1]
5
```

8.4.128 @MD5

String mode: @**MD5[***s,string***]**
File mode: @**MD5[***[f,]filename***]**

Returns the 32 hexadecimal digit MD5 hash of the character in ***string*** or of the contents of the file ***filename***. The first parameter must be ***s*** for a string, and any leading or trailing whitespace characters in ***string*** are included.

Filename may be specified with or without an optional ***f***. @MD5 returns **-1** if the file does not exist, or it cannot be read.

Since ***4NT*** and ***TC*** handle all internal strings as Unicode, @MD5 will return different results for a string and the identical string in an ASCII file.

See also: @SHA256 490, @SHA384 490, @SHA512 490, and @CRC32 443.

8.4.129 @MIN

@**MIN[***a,b,c,...***]** : Returns the smallest in the list of parameters. All parameters must be integers in the range **-2147483647** `to` **2147483647** and must be separated either by whitespace or by commas.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

Example:

```
echo %@min[1,5,2,0,-1]
-1
```

8.4.130 @MONTH

@**MONTH[***date[,format]***]** : Returns the month number for the specified date (1-12). See date formats 72 for information on acceptable date formats.

@MONTH accepts an optional second parameter specifying the date format:

0 default
1 USA (mm/dd/yy)
2 Europe (dd/mm/yy)
3 Japan (yy/mm/dd)
4 ISO (yyyy/mm/dd)

Examples:

```
echo %@month[01-01-1980]
echo %@month[%_date]
```

8.4.131 @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 424 for additional details.

Note: The @NAME function makes no assumption about the existence of a file or directory. Its ***filename*** parameter can be any string and the function will attempt to extract from it a base name.

Examples:

```
echo %@name[xyz.abc]
echo "%@name[%_comspec]"
```

8.4.132 @NUMERIC

@NUMERIC[*[+|-]string***]** : Returns **1** if ***string*** is numeric, and **0** otherwise.

To be numeric, all of the following must be true:

1. the first character may be a + or - sign,
2. the rest of ***string*** must be composed entirely of decimal digits (0 to 9), and the thousands (ThousandsChar 147) and decimal (DecimalChar 113) separators.
3. If the first character (second character if the first one is a sign) is the decimal separator, only decimal digits may follow it.
4. If the first character (second character if the first one is a sign) is not the decimal separator, it must be a decimal digit, and multiple thousands and decimal separators may be present.

Examples:

function	***value***

%@numeric[12345]	1
%@numeric[-12345]	1
%numeric[.12345]	1
%@numeric[$12.34]	0
%@numeric[5.00.125]	1
%@numeric[+5.00.125,5]	1
%@numeric[.00.125]	0
%@numeric[-5,.00.125]	1

8.4.133 @OPTION

@**OPTION[***directive***]** : Returns the current value of the requested .INI file 91 directive. All directives which can be altered dynamically are supported. If ***directive*** is not supported, an error occurs.

For Configuration Directives 99, the current value returned may not match that stored in the *.INI* file.

For Color Directives 98, the current value is returned as a single number (0-255) combining foreground and background specifications. See Colors, Color Names & Codes 552 for details.

Examples:

```
echo %@option[passiveftp]
echo %@option[stdcolors]
```

8.4.134 @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 466 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 424 for additional details.

Note: The @PATH function makes no assumption about the existence of a file or directory. Its ***filename*** parameter can be any string, and the function will attempt to remove from it a "base name".

Examples:

```
command                                          result

echo "%@path["c:\program files\xyz.abc"] "       "c:\program files\"
echo "%@path[xyz.abc]"                          ""
echo "%@path[%_comspec]"                  "c:\jpsoft\rlsdu\"
```

8.4.135 @OWNER

@**OWNER[***filename***]**: Returns the owner of the specified file (if any).

8.4.136 @PERL

@**PERL[***expression***]** : Executes the specified Perl expression (requires Active State Perl 5.8).

Note: Because of a number of serious bugs in the Windows implementation of embedded Perl 5.8, the Perl support in ***4NT*** / ***TC*** is unofficial and unsupported. Use at your own risk!

8.4.137 @PING

@**PING[***host[,timeout[,packetsize]]***] :** Returns the response time in milliseconds for the specified host. ***Host*** is the IP address or name, ***timeout*** is the maximum number of seconds to wait, and ***packetsize*** is the size of the data packet sent to the host in the ping request. The ***timeout*** defaults to 5 seconds, and ***packetsize*** defaults to 64 bytes. The minimum packet size is 8 bytes and the maximum is 2048 bytes.

A negative value indicates an error. If the request times out, @PING returns **–1**. An unreachable host returns **-2**. An invalid address returns **-3**.

Examples:

```
echo %@ping[microsoft.com]
echo %@ping[microsoft.com,10]
echo %@ping[microsoft.com,,16]
echo %@ping[192.168.01,2,512]
```

8.4.138 @QUOTE

@**QUOTE[***string***]** : Returns a double quoted argument if it contains any whitespace characters.

8.4.139 @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 the command processor starts and will therefore produce a different sequence of numbers each time you use it.

Examples:

```
echo %@random[0,1]
echo %@random[-10,10]
```

8.4.140 @READSCR

@**READSCR[***row,col,length***]**: Returns the text displayed in the command processor window at the specified location. The upper left corner of the window is location 0,0. The ***row*** and ***column*** can be specified as an offset from the current cursor location by preceding either value with a [**+**] or [**-**]. For example:

```
%@readscr[-2,+2,10]
```

returns 10 characters from the screen, starting 2 rows above and 2 columns to the right of the current cursor position.

8.4.141 @READY

@**READY[***d:***]**: Returns **1** if the specified drive is ready; otherwise returns **0**. The drive letter must be followed by a colon.

Examples:

```
echo %@ready[E:]
echo %@ready[%_boot:]
```

8.4.142 @REGCREATE

@**REGCREATE[***HKEY...\subkey***]**: Create a new registry subkey. The parameter starts with the root key, which can be abbreviated:

Full root key	Short
HKEY_CLASSES_ROOT	HKCR
HKEY_CURRENT_USER	HKCU
HKEY_LOCAL_MACHINE	HKLM
HKEY_USERS	HKU
HKEY_CURRENT_CONFIG	HKCC

The remainder of the parameter (after the backslash) specifies the new subkey. The entire name must be quoted if it contains any white space or special characters, for example:

```
@REGCREATE["HKLM\Software\My Company\My Product\User"]
```

REGCREATE will create any intermediate keys necessary. For example, `@REGCREATE[HKCU\key1\key2\key3]` will create all three keys (if they do not already exist). REGCREATE returns **0** if the subkey was created or the Windows error number if an error occurred.

See also: @REGQUERY 485 (read a value), @REGSET 485 (write a value), and @REGSETENV 486 (write and broadcast a value).

8.4.143 @REGDELKEY

@**REGDELKEY[***HKEY...\key***]** : Deletes the specified key and all of its subkeys. Returns **1** if the key was deleted, **0** otherwise. The expression must be enclosed in double quotes if it contains any separator characters (space, comma, or tab).

Note: use EXTREME caution with this function. It has the potential for causing irreparable damage to your registry and can even prevent Windows from booting!

See @REGCREATE 484 for information on the format of the key name.

8.4.144 @REGEX

@**REGEX[***expression,string***]** : Returns the number of matching groups in the string. If no groups are specified, it returns **1** if the ***expression*** was found and **0** if it was not. The expression must be enclosed in double quotes if it contains any separator characters (space, comma, or tab). See Regular Expression Syntax 526 for supported expressions.

8.4.145 @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 526 for supported expressions. (This function is basically a wildcard-enabled @INDEX.)

8.4.146 @REGEXIST

@**REGEXIST[***HKEY...\key***]** : Returns **1** if the specified key exists, **0** otherwise

The expression must be enclosed in double quotes if it contains any separator characters (space, comma, or tab).

See @REGCREATE 484 for information on the format of the key name.

8.4.147 @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 526 for supported expressions.

8.4.148 @REGQUERY

@**REGQUERY[***HKEY...\subkey\value***]:** Read a value from the registry. REGQUERY supports keys of type REG_DWORD, REG_QWORD, REG_EXPAND_SZ, REG_SZ, REG_DWORD_LITTLE_ENDIAN , and REG_QWORD_LITTLE_ENDIAN. If the key is of type REG_EXPAND_SZ, the value is returned without further expansion. If the value name does not exist, the function returns **–1**. If the value name is not supplied, REGQUERY returns the unnamed value for the specified key (the first value with a NULL name). To retrieve an unnamed value, add a trailing \ to the name.

Note: Remember to use quotes around any entry containing spaces or commas!

See @REGCREATE 484 (create a subkey) for information on the format of the key name. See also: @REGSET 485 (write a value) and @REGSETENV 486 (write and broadcast a value).

8.4.149 @REGSET

@**REGSET[***HKEY...\subkey\value,type,data***]:** Write a value to the registry. REGSET supports keys of type REG_DWORD, REG_SZ, REG_EXPAND_SZ, and REG_DWORD_LITTLE_ENDIAN. ***Type*** is the value type (REG_DWORD, REG_EXPAND_SZ, or REG_SZ). ***Data*** is the data to set. If this parameter is not supplied, @REGSET will remove the value. REGSET returns **0** if the value was written or the Windows error number if an error occurred.

Note: Remember to use quotes around any entry containing spaces or commas!

See @REGCREATE 484 for information on the format of the key name. See also: @REGQUERY 485 (read a value) and @REGSETENV 486 (write and broadcast a value).

8.4.150 @REGSETENV

@**REGSETENV[***HKEY...\subkey\value,type,data***]** : The same as @REGSET 485, but a broadcast message is sent to all applications when the change is made, so that any application monitoring such messages can respond to the change immediately if it is designed to do so. @REGSETENV returns **0** if the value was written or the Windows error number if an error occurred.

Note: Remember to use quotes around any entry containing spaces or commas!

See @REGCREATE 484 for information on the format of the key name. See also: @REGQUERY 485 (read a value) and @REGSET 485 (write a value).

8.4.151 @REMOTE

@**REMOTE[***d:***]**: Returns **1** if the specified drive is a remote (network) drive; otherwise returns **0**. The drive letter must be followed by a colon.

Examples:

```
echo %@remote[e:]
echo %@remote[%_disk]
```

8.4.152 @REMOVABLE

@**REMOVABLE[***d:***]**: Returns **1** if the specified drive is removable (e.*g.* floppy disk, removable hard disk, USB storage device, etc.), **0** otherwise. The drive letter must be followed by a colon.

Examples:

```
echo %@removable[e:]
echo %@removable[%_disk]
```

8.4.153 @REPEAT

@**REPEAT[***char,count***]** : Returns the character ***char*** repeated ***count*** times (***count*** may not exceed 8,191).

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

Examples:

```
function                    value
%@repeat[%@char[95],10]     __________
7%@repeat[ ,7]spaces        7       spaces
%@repeat[x,10]              xxxxxxxxxx
```

8.4.154 @REPLACE

@**REPLACE[***string1, string2, text***]**: Replaces all occurrences of ***string1*** in the ***text*** string with ***string2***. For example, **%@replace[w,ch,warming]** returns the string "charming".

The search is case sensitive.

Examples:

```
echo %@replace[\,/,"ftp:\\server\etc"]
echo %@replace[%=,,,A better, command processor]
```

8.4.155 @REVERSE

@**REVERSE[***string***]** : Reverses the order of the characters in ***string***.

8.4.156 @REXX

@**REXX[***[=]expr***]**: Calls the REXX interpreter to execute the expression. Returns the numeric code or string result from REXX. Console output from the REXX interpreter is suppressed while executing the expression. Note that the command processor expands variables and functions before passing ***expr*** to REXX.

Examples:

```
echo %@rexx[ = 3 * 4 ]
set myprog=d:\path\xyz.exe
echo %@rexx[ address(%@name[%myprog]); return address() ]
```

Note: This function requires that a recognized REXX interpreter be installed and properly configured. See REXX Support 175 for more information on the REXX language.

8.4.157 @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

8.4.158 @RTRIM

@**RTRIM[***string1,string2***]**: - Returns ***string2*** with all the trailing characters in ***string1*** removed. ***String1*** must be enclosed in double quotes if it contains any spaces, tabs, or commas.

Examples:

```
echo "%@rtrim[985NTXP,Windows XP]"
"Windows "
```

8.4.159 @RUBY

@**RUBY[***expression***]** : Returns the string result of the Ruby expression. Note that the Ruby environment is persistent within ***4NT*** and ***TC***, 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 421 internal variable, and the value returned by the last @RUBY call with the RUBYVALUE 421 internal variable.

8.4.160 @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 343 command.

8.4.161 @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 533 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 19 are used in the ***program***, @SEARCH will search for the first program file that matches the wildcard specification, and returns the drive and path for that file plus the wildcard filename (*e.g.*, *E:\UTIL*.COM*).

Program and each directory specification in ***path*** must be in quotes if they contain white space or special characters.

@SEARCH accepts an optional third parameter specifying whether to search the current directory. If ***n*** is 0, @SEARCH will not look for the file in the current directory. If ***n*** is 1 (the default), @SEARCH will look in the current directory before searching the path.

Examples:

```
echo %@search[notepad]
echo %@search[msv*.dll,"d:\my dir\"]
```

8.4.162 @SELECT

@SELECT[*filename,top,left,bottom,right,title[,1]***]**: Pops up a selection window with the lines from the specified file, allowing you to display menus or other selection lists from within a batch file. You can move through the selection window with standard popup window navigation keystrokes, including character matching (see Popup Windows 536 for details; to change the navigation keys see Key Mapping directives 102).

Filename must be in quotes if it contains white space or special characters. The file size is limited only by available memory. To select from lines passed through input redirection or a pipe, use **CON:** as ***filename***. To select from lines in the Windows clipboard, use **CLIP:** as ***filename***.

If you use the optional 7th parameter **1** (immediately after the window title), the list will be sorted alphabetically.

Return value:

- the text of the line the scrollbar is on if you press ***Enter***
- an empty string if you press ***Esc***.

Examples:

```
call %@select["d:\path\my menu.txt",5,10,15,40,Select an option]
help %@word["=",0,%@select[d:\path\4nt.ini,0,0,10,40,Select a
directive]]
```

8.4.163 @SERIAL

@SERIAL[*d:***]**: Returns the serial number of the specified disk drive (in hex, i.e.: ABCD:0123). The drive letter must be followed by a colon.

Examples:

```
echo %@serial[C:]
echo %@serial[%_disk:]
```

See also: @LABEL 476.

8.4.164 @SFN

@SFN[*filename***]**: Returns the fully expanded short ("8.3") filename for a long ***filename***. The ***filename*** may contain any valid filename element including drive letter, path, filename and extension. The entire name including all intermediate paths will be returned in short name format. If ***filename*** does not refer to an actual file, the results are unpredictable.

Examples:

```
echo %@sfn["c:\program files\xyz.abc"]
echo %@sfn[%_comspec]
```

8.4.165 @SHA1

@**SHA1[***filename***]** : Returns the SHA1 checksum of the specified file.

See also @SHA256 490, @SHA384 490, @SHA512 490, @MD5 480, and @CRC32 443.

8.4.166 @SHA256

@**SHA256[***filename***]** : Returns the SHA2-256 checksum of the specified file.

See also @SHA384 490, @SHA512 490, @MD5 480, and @CRC32 443.

8.4.167 @SHA384

@**SHA384[***filename***]** : Returns the SHA2-384 checksum of the specified file.

See also @SHA256 490, @SHA512 490, @MD5 480, and @CRC32 443.

8.4.168 @SHA512

@**SHA512[***filename***]** : Returns the SHA2-512 checksum of the specified file.

See also @SHA256 490, @SHA384 490, @MD5 480, and @CRC32 443.

8.4.169 @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.

8.4.170 @SNAPSHOT

@**SNAPSHOT[***DESKTOP* | *window[,n]***]** : Save the desktop or a specific window to the clipboard as a BMP. The window argument can be either **DESKTOP** or a window title (which can include wildcards). The optional second argument specifies whether you want only the client area (0) or the entire window (1) to be saved. Returns **0** if successful.

8.4.171 @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]

8.4.172 @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***.

8.4.173 @SUBSTR

@**SUBSTR[***string,start,length***]** : An older version of @INSTR 473. 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 EscapeChar 117. The quotes count in calculating the position of the substring. @INSTR, which has ***string*** as its last parameter, does not have this restriction.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

Examples:

```
echo %@substr[this is useful,8]
echo %@substr[this is useful,8,-2]
echo %@substr["commas, they DO matter",9]
echo %@substr[commas%=, they DO matter,9]
```

See also: @INSTR 473.

8.4.174 @SUBST

@**SUBST[***n, string1, string2***]**: Substitutes ***string1*** starting at position ***n*** in ***string2***.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

8.4.175 @TIME

@**TIME[***hh:mm:ss***]** : Returns the number of seconds since midnight for the specified time. The time must be in 24-hour format. "am" and "pm" are ignored. Any non-numeric character, except a right bracket **]** can be used to separate the hour, minute and second subfields.

Examples:

```
echo %@time[12:34:56]
echo %@time[%_time]
```

8.4.176 @TIMER

@**TIMER[***n[,precision]***]** : Returns the current split time for a stopwatch started with the TIMER 379 command. The value of ***n*** specifies the timer to read and can be 1, 2, or 3.

@**TIMER** accepts an optional second argument to return the timer split as a floating-point numeric value suitable for arithmetic. The possible values are:

s split time in seconds (2 digit decimal precision)
m split time in minutes (4 digit decimal precision)
h split time in hours (5 digit decimal precision)

8.4.177 @TRIM

@**TRIM[***string***]** : Returns the string with the leading and trailing white space (space and tab characters) removed.

8.4.178 @TRUENAME

@**TRUENAME[***filename***]** : Returns the true, fully-expanded name for a file. @TRUENAME will "see through" junctions, a SUBST or network mapping. Wildcards cannot be used in the filename.

Note: The @TRUENAME function makes no assumption about the existence of a file or directory. Its ***filename*** parameter can be any string and the function will attempt to turn it into a fully qualified "volume + path + name" specification, whether that full reference exists or not.

filename must be in quotes if it contains white space or special characters.

8.4.179 @TRUNCATE

@**TRUNCATE[***handle***]** : Truncate the file opened for write access by @FILEOPEN 459 at the current position of the file pointer, where ***handle*** is the value returned by @FILEOPEN 459.

See also the related handle-based functions:

@FILECLOSE 458 Close a file handle
@FILEOPEN 459 Open a file handle
@FILEREAD 460 Read next line from a file handle
@FILESEEK 461 Move a file handle pointer
@FILESEEKL 461 Move a file handle pointer to a specified line
@FILEWRITE 463 Write next line to a file handle
@FILEWRITEB 463 Write data to a file handle

8.4.180 @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).

8.4.181 @UNICODE

@**UNICODE[***string***]** : Returns the space separated list of the Unicode values of the characters in ***string***. You can use the EscapeChar 117 before a special character (i.e., a quote or greater than (>) sign) in ***string***.

See also: @ASCII 438.

Examples:

function	***value***
%@unicode[a]	97
%@unicode[A]	65
%@unicode[%=`]	96
%@unicode[abc]	97 98 99

8.4.182 @UNIQUE

@**UNIQUE[***path***]** : Creates a zero-length file with a unique name in the specified directory, and returns its the full name and path. If no ***path*** is specified, the file will be created in the current directory. The file name will be FAT-compatible regardless of the type of drive on which the file is created. This function allows you to create a temporary file without overwriting an existing file.

The ***path*** must be in quotes if it contains white space or special characters.

Because the file is created, if NoClobber 133 is in effect, 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 229, DELAY /M, or BEEP 208 (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.

8.4.183 @UNQUOTE

@**UNQUOTE[***string***]** : Returns the argument with all double quotes removed.

8.4.184 @UNQUOTES

@**UNQUOTES[***string***]** : Returns the argument with leading and trailing double quotes removed.

8.4.185 @UPPER

@**UPPER[***string***]** : Returns ***string*** converted to upper case.

8.4.186 @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, **4NT.EXE** returns values for:

```
CompanyName
FileDescription
FileVersion
InternalName
LegalCopyright
LegalTrademarks
OriginalFilename
ProductName
ProductVersion
Build
```

To return **CompanyName** :

```
echo %@verinfo[4nt.exe,companyname,040904E4]
```

Note: Most, but not all, executables under Windows contain a **FileVersion** field. The number, names and contents of the specific information fields and language/codepage pairs provided within a given application can potentially be anything the programmer decided to use.

8.4.187 @WATTRIB

@**WATTRIB[***filename[,-attributes[,p]]***]**: This function is similar to @ATTRIB 439, but supports file selection based on the following extended attributes available on NTFS volumes.

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

See also: Attributes Switches 28 and the ATTRIB 198 command

8.4.188 @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[19]. 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

8.4.189 @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 440.

8.4.190 @WINCLASS

@WINCLASS[*classname***] :** Returns the window title of the first window with the specified class name, or an empty string if no windows match.

8.4.191 @WINEXENAME

@WINEXENAME[*title***]**: Returns the executable name for the first window matching ***title*** (which can include wildcards 19), or an empty string if none.

8.4.192 @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 **1** MIPS **2** ALPHA **3** PPC **4** SHX **5** ARM **6** IA64 **7** ALPHA64
2	Processor bit mask (set of configured processors)
3	Number of processors
4	Type of processor **586** Pentium:
5	Processor level
6	Processor revision
7	page size, bytes
8	virtual memory allocation granularity, bytes

8.4.193 @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

8.4.194 @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
1	Height of screen
2	Width of arrow bitmap on horizontal scroll bar
3	Height of arrow bitmap on horizontal scroll bar
4	Height of title bar
5	Width of window border
6	Height of window border
7	Width of dialog box border
8	Height of dialog box border
9	Height of thumb box on vertical scroll bar
10	Width of thumb box on horizontal scroll bar
11	Width of icon
12	Height of icon
13	Width of cursor
14	Height of cursor
15	Height of single line menu bar
16	Width of client area for full-screen window
17	Height of client area for full-screen window
18	Height of Kanji window
19	Mouse present flag **0** no **1** yes
20	Height of arrow bitmap on vertical scroll bar
21	Width of arrow bitmap on vertical scroll bar
22	Debug version of Windows **0** no **1** yes
23	Left and right mouse buttons swapped **0** no **1** yes

28	Minimum width of a window
29	Minimum height of a window
30	Width of bitmaps in title bar
31	Height of bitmaps in title bar
32	Width of window frame that can be sized
33	Height of window frame that can be sized
34	Minimum tracking width of window
35	Minimum tracking height of window
41	Is Pen Windows installed? **0** no **1** yes
42	Is DBCS version of USER.EXE installed? **0** no **1** yes
43	Number of buttons on mouse
70	Windows will display visual info in place of audible info **0** no **1** yes
73	Computer has a slow processor **0** no **1** yes
74	Is Windows set up for Arabic/Hebrew? **0** no **1** yes
75	Mouse has a wheel **0** no **1** yes
76	Coordinate of left side of virtual screen
77	Coordinate of top of virtual screen
78	Width in pixels of virtual screen
79	Height in pixels of virtual screen
80	Number of monitors on desktop

8.4.195 @WINPOS

@**WINPOS[***title***]**: Returns the screen coordinates of the window with the specified title, in the format "*top,left,bottom,right*".

8.4.196 @WINSTATE

@**WINSTATE[***title***]** : Returns the window state of the first window matching ***title*** (which can include wildcards 19). The return values are:

Value	*Window state*
0	Hidden
1	Normal
2	Minimized
3	Maximized

8.4.197 @WINSYSTEM

@**WINSYSTEM[***n[,v]***]**: Sets or returns the value of the requested Windows system-wide parameters.

To retrieve a parameter, the format is **%@winsystem[n]** where ***n*** is the appropriate **GET** number from the table below.

To set a parameter, the format is **%@winsystem[n,v]** where ***n*** is the appropriate **SET** number from the table below and ***v*** is the desired new value for that parameter.

Where the selection is a state, the legal values are **0** for off/disabled, and **1** for on/enabled.

Where the selection is a width or height, the values are in pixels.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

Note: This function provides direct access to the **SystemParametersInfo** API. Not all available parameters are listed here and your Windows configuration may support additional parameters. See your Windows technical documentation for details, and use with caution.

GET	*SET*	*Parameter to GET or SET*
1	2	Beep state
5	6	Border width
10	11	Keyboard repeat speed (0 to 31)
13	13	Width of an icon cell
14	15	Screen saver time-out (seconds)
16	17	Screen saver state
22	23	Keyboard repeat delay setting (0-3).
24	24	Height of an icon cell
25	26	Icon title wrapping state
27	28	Pop-up menu alignment
37	38	Full-window dragging state
56	57	Show Sounds accessibility flag
68	69	Keyboard preference state (0=mouse, 1=keyboard)
70	71	Screen reviewer utility state
74	75	Font smoothing feature state
79	81	Time-out for the low-power phase of screen saving (seconds)
80	82	Time-out value for the power-off phase of screen saving (seconds)
83	85	Low-power phase of screen saving state
84	86	Power-off phase of screen saving state
89	90	Locale identifier for the system default input language.
93	94	Mouse Trails feature state. (0 or 1= disabled, >1= number of cursors in the trail)
95	95	Snap-to-default-button feature state
98	99	Width of the mouse pointer WM_MOUSEHOVER message trigger rectangle
100	101	Height of the mouse pointer WM_MOUSEHOVER message trigger rectangle
102	103	Time in the hover rectangle for the mouse pointer to trigger a WM_MOUSEHOVER message (milliseconds)

104	105	Number of lines to scroll when the mouse wheel is rotated
106	107	Time that the system waits before displaying a shortcut menu when the mouse cursor is over a submenu item (milliseconds)
110	111	IME status window state - per user (0=invisible, 1=visible)
112	113	Current mouse speed (1 to 20).
4096	4097	Active window tracking state
4098	4099	Menu animation feature state.
4100	4101	Combo box animation state.
4102	4103	List box smooth-scrolling effect state.
4104	4105	Gradient effect for window title bars.
4106	4107	Menu access keys underline state.
4108	4109	Active window tracking Z-order state.
4110	4111	Hot-tracking state.
4114	4115	Menu fade animation state.
4116	4117	Selection fade effect state.
4118	4119	ToolTip animation state.
4120	4121	Type of ToolTip animation (1 for fade, 0 for slide)
4122	4123	Cursor shadow state.
4124	4125	State of the Mouse Sonar feature
4126	4127	Mouse clicklock state
4128	4129	Mouse vanish feature state
4130	4131	***(WinXP/2003/Vista)*** Whether native User menus have flat menu appearance.
4132	4133	***(WinXP/2003/Vista)*** 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	***(Win/XP/2003/Vista)*** Time delay before the primary mouse button is locked.
8202	8203	***(WinXP/2003/Vista)*** Type of font smoothing (32769=standard anti-aliasing, 32770=ClearType).
8204	8205	***(WinXP/2003/Vista)*** Contrast value used in ClearType smoothing (1000-2200)
8206	8207	***(WinXP/2003/Vista)*** Width of the left and right edges of the focus rectangle
8208	8209	***(WinXP/2003/Vista)*** Height of the top and bottom edges of the focus rectangle

.

GET	SET	Parameter to GET or SET

8.4.198 @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 394.

Examples:

%@wmi[root\cimv2,"SELECT name FROM Win32_Processor"]

%@wmi[root\cimv2,"SELECT name, state FROM Win32_service",4]

8.4.199 @WORD

@**WORD[***["sep_list",]n,string***]** : Returns the ***n***th word in ***string***. The first (leftmost) word is numbered **0**. If ***n*** is negative, words are counted backwards from the end of ***string***, and the absolute value of ***n*** is used. You can specify the rightmost word by setting ***n*** to **-0**.

You can specify a range of words to return with the syntax:

@**WORD**[["sep_list",]start[-end | +range],string]

Specify an inclusive range with a **-**. For example:

%@word[2-4,A B C D E F G] will return "C D E". (Note that you cannot use inclusive ranges when starting from the end.)

You can specify a relative range with a **+**. For example:

%@word[2+1,A B C D E F G] will return "C D".

The default list of separators for @FIELD 456, @FIELDS 457, @WORD 501 and @WORDS 502 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 117, e.g., %=". Alphabetic characters in sep_list are case sensitive.

@FIELD 456 and @FIELDS 457 differ from @WORD 501 and @WORDS 502 in how multiple consecutive separators are counted. @WORD 501 and @WORDS 502 consider a sequence as a single separator, and ignore separators at either end of string. In contrast, @FIELD 456 and @FIELDS 457 count each occurrence of a separator individually, including those at either end of string.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits). To use hexadecimal form for a negative ***n***, remember to use 32-bit 2's complement arithmetic, e.g., **0xFFFFFFFF** for **-1**.

If ***string*** is double quoted, you must specify ***sep_list***.

See also: @WORDS 502, @FIELD 456, @FIELDS 457.

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

8.4.200 @WORDS

@**WORDS[***"sep_list",]string***]**: Returns the number of words in ***string***.

The default list of separators for @FIELD 456, @FIELDS 457, @WORD 501 and @WORDS 502 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 117, e.g., %=". Alphabetic characters in ***sep_list*** are case sensitive.

@FIELD 456 and @FIELDS 457 differ from @WORD 501 and @WORDS 502 in how multiple consecutive separators are counted. @WORD 501 and @WORDS 502 consider a sequence as a single separator, and ignore separators at either end of ***string***. In contrast, @FIELD 456 and @FIELDS 457 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 501, @FIELD 456, @FIELDS 457.

8.4.201 @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.

8.4.202 @XMLPATH

@**XMLPATH[***filename,path***]** : Return the text of the current element.

filename - name of XML file
path - one or more element accessors separated by a /.

8.4.203 @YEAR

@**YEAR[***date[,format]***]**: Returns the year for the specified date. See date formats 72 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)

9 Setup & Troubleshooting

- Registration 503
- Troubleshooting Service and Support 503
- Supported Platforms 506

9.1 Registration

There are no separate **trial** and **registered** versions of our products. Without registration, a trial version is fully functional for 30 days of use.

At any time you can apply your current personal registration information to a trial version in order to turn it into a registered product. Use the command VER /R 389 from the prompt to verify the status of the copy you are currently running. (Under ***TC***, you can also view the Help/About 78 menu entry.)

When you purchase a new or upgrade copy of ***4NT*** or ***TC***, you will receive an email with your name and registration key. Start ***4NT*** or ***TC***, type **Option** at the command line and select the **Register** tab, and enter the registration information exactly as you received it in the email. Remember to save your registration key in a safe place in case you need to reinstall. If you have lost your registration key, you can request a replacement by contacting JP Software at sales@jpsoft.com, or at one of the addresses listed at the start of this file.

9.2 Troubleshooting, Service & Support

If you need help with ***4NT*** or ***TC***, we encourage you to review our documentation and then contact us for assistance if required.

If you need help with sales, ordering, or shipments (including defective disks or other materials which were shipped to you), or with registration codes, please contact our Sales and Customer Service department. See Contacting JP Software 505 for our email address, mail address, and telephone numbers. Note that Sales and Customer service staff cannot assist you with technical problems and conversely Technical Support representatives cannot answer your sales or registration questions.

If you need technical support for the command processor, review the Technical Support 503 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.

9.2.1 Technical Support

Support Plans

Standard, no-charge support is available electronically through our Support Forum 503 (see below). We also offer a paid support option which includes automatic upgrades and support by private email or telephone. For complete details on all support options, including plans currently offered and support terms and conditions, see our web site at http://jpsoft.com/.

Before you contact Technical Support, please review the What Information do we need? 504 section which outlines the basic data we need to best address your questions and concerns.

Online Support

The primary venue for Technical Support is via our free online Support Forum, where our support personnel can read and respond to your messages, and other users can participate in and benefit

from the exchange. The Forum is a lively community frequented by a number of experienced and helpful users. JP Software representatives read every Forum message and respond as promptly as reasonably possible whenever appropriate.

If you have any kind of Internet access, even if only email, chances are you can use the Forum which we make accessible as a mailing list, a news group, and a set of web pages. Forum members must provide a valid email address and a full name to be able to post, but you do 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 http://jpsoft.com/.

A number of other support resources are available from our web site, including documentation files, technical tips and discussions, other technical information, and links to other sites. We update this information regularly, and we encourage you to check the Technical Support area of the web site to see if the information there will address any questions you have.

If you are unable to gain access to the forum, or you need to include confidential information in your support request, contact us via email at support@jpsoft.com and we will assist you in resolving the problem with forum access, or assist you with your request privately if appropriate. Please do not use that address for standard support questions which can be posted on the forum.

If you are a paid support customer you should use the online Support Forum for routine questions. To create a private support incident refer to the materials sent to you with your subscription for contact information, or email priority_support@jpsoft.com and include your support ID (mail to this address may not be answered if it does not include a valid support ID).

Note that for security and "anti-spam" reasons, the support address filters out all non-text data (including screen shots and others) and such prohibited material will probably be lost before it ever reaches us. Technical support messages should be sent as standard ASCII text. Please do not transmit attached files, binary files, screen images, or any file over 10K bytes in size to any of our electronic technical support addresses unless asked to do so by our support staff. We will be unable to respond to (and may not even receive) messages that do not met these basic criteria.

What Information do we need?

Before contacting us for support, please check this help file and other documentation for answers to your question. If you can't find what you need, try the Index. If you're having trouble getting the command processor to run properly, review the information on Error Messages 507, and look through the Support Forum for any last-minute information.

If you need help with sales, ordering, shipments, registration keys, or other similar non-technical issues please contact our Sales and Customer Service department. Technical Support will not be able to assist you with those matters. Conversely, Customer Service is not equipped to answer your technical questions. See Contacting JP Software 505 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 389 command to determine the command processor version and operating system version. This item is essential**!** Every question posted on the Forum should include a brief identification such as "4NT 7.01 build 335 under

XP+SP2" or something similar.

- **What exactly did you do?** A concise description of what steps you must take to make the problem appear is much more useful than a long analysis of what might be happening. In most cases, posting the exact command line(s) giving you trouble is the simplest approach.

- **What did you expect to happen?** Tell us the result you expected from the command or operation in question, so that we understand what you are trying to do. Something that seems "obvious" to you might not be so to others. For example, tell us "I was expecting the file name to be in upper case" or a similar brief explanation.

- **What actually happened?** At what point did the failure occur? If you saw an error message or other important or unusual information on the screen, what **exactly** did it say? Don't simply tell us "it didn't work". For example, if you were expecting output from a command and saw none, at least tell us that much.

- **Briefly, what techniques did you use to try to resolve the problem?** What results did you get? One technique that tends to solve many problems is to review the help for the command or feature in question and try it with the documented exact correct syntax, as opposed to some undocumented alternative.

- **If the problem seems related to startup and configuration issues,** what are the contents of any startup files you use (such as TCSTART 8 or 4START 8, TCEXIT 8 or 4EXIT 8, and the .INI file 91), any batch files they call, and any alias or environment variable files they load? You do not need to include the entire file, of course, but at least any entry you think might be relevant, such as a specific INI directive or a command.

- **Can you repeat the problem or does it occur randomly?** If it's random, does it seem related to the programs you're using when the problem occurs? Random or occasional problems are very difficult to diagnose. Do your best to determine some sort of pattern or sequence of events that triggers the problem. If you can't reproduce it, chances are we won't be able to either. Note that mysterious unexplainable problems often permanently disappear after simply reloading the program or even rebooting the system.

- If the command processor experiences a fatal failure, such as a Windows General Protection Fault (GPF), Unhandled Exception, or similar unrecoverable error, it will attempt to dump the relevant memory locations and other useful data to a text file (**4NT.GPF** or **TCMD.GPF**) into its directory if at all possible. That *.GPF file may be important for tracking down the exact internal location of the failure, whether in the command processor executable itself or (most often) in some Windows component that it happened to invoke, and Technical Support may ask you to send a copy of that text file to a specified address or post its contents on the Forum.

9.2.2 Contacting JP Software

You can contact JP Software at the following addresses and numbers. Our normal business hours are 9:00 AM to 5:00 PM weekdays, Eastern US time (except holidays).

Address:	**JP Software Inc.** **P.O. Box 328** **Chestertown, MD 21620** **USA**
Main number:	**(410) 810-8818**
Fax:	**(410) 646-0026**

Order Line:	**(410) 810-8819**
Online:	Web site: http://jpsoft.com/ FTP site: ftp://jpsoft.com Sales / Customer Service: sales@jpsoft.com
Technical Support:	Standard (no-charge) support: Available via our online **Support Forum**, accessible from the support area of our web site.

See Technical Support 503 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.

9.3 Supported Platforms

Take Command is a Win32 graphical application.

4NT is a Win32 console application

Both are designed to run under Windows 2000, Windows XP, Windows 2003, and Windows Vista.

9.4 Versions

4NT and ***TC*** are available only in **Unicode** versions, designed for use under Windows 2000, XP, 2003, and Vista. The trailing **U** of the internal variable _4VER 410, which had been used in versions 5 and 6, has been dropped.

Note: To properly view Unicode glyphs, you must use a suitable Unicode font. In most Windows configurations, the Microsoft-provided ***Lucida Console*** will be an acceptable starting point but it only supports a limited subset of common Unicode glyphs. Some Unicode material may require a more complete font. The phrase **Unicode support** in Windows terminology refers to what is technically named **UTF-16LE** encoding. The current versions of ***4NT*** and ***TC*** can handle all 65535 Unicode glyphs, and can create either ASCII or Unicode output.

9.5 Help File

The distribution packages for ***4NT*** and ***TC*** contain **jphelp.chm,** a help file in Window's compiled HTML format. That file includes description and syntax for all commands, variables and functions, as well as reference information to assist you in installing and using the command processor and developing batch files, aliases and functions. The command processor uses the default Windows help system to display the contents of the jphelp.chm. Under most configurations, Windows will remember the last used settings (window size, tab selected, etc.) for that file. Once you've started the help system with F1 120 or the HELP 284 command, you can use standard Windows HTML Viewer (HH.EXE) keystrokes to navigate. For more information, see your Windows documentation.

The same material is also available in alternative formats such as PDF. Please check our web site regularly to make sure you have the most recent version of the help files, since they are usually updated more frequently than the programs.

You can request help at the prompt from the Help menu ***(TC)***, by typing HELP 284 (or HELP 284 plus a

command name), or by pressing the F1 120 key at any time when the command processor is accepting keyboard input at the prompt. If you use the HELP 284 command by itself you will be taken to an introductory page, but if you follow the command with a topic name (e.g. **help copy**) you will see help on the requested topic if available.

If you type a command name on the line and then press F1 120, the help system will provide context sensitive help by using the first word on the line as the help topic. For example, if you press **F1** after entering each of the command lines shown below you will get the display indicated:

`[c:\]`	Overview
`[c:\] copy *.* a:`	Help on the COPY command
`[c:\] c:\util\map`	"The page cannot be displayed"

You can use this feature to obtain help directly on any topic, not just on commands. All internal commands and most internal variables, functions and directives have their own topic, allowing you to directly query, for example, **help @eval** (help for the @eval[] function) or **help _disk** (help for the _DISK internal variable).

You can also invoke help for the word immediately above (or immediately to the left of) the cursor by pressing the Ctrl-F1 120 key. This feature is especially useful when you need the syntax for a variable function.

If the topic you seek is not listed, look for a suitable cross reference from the **Index** tab or use the **Search** tab. The topics you use most often can be stored and recalled through the **Favorites** tab.

Quick Help:

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

```
copy /?
```

Then you will see "quick help" for that command, displayed in the command window.

9.6 Error Messages

This section lists error messages generated by the command processor, and includes a recommended course of action for most errors. If you are unable to resolve the problem after reviewing these help files, contact JP Software for technical support 503.

Error messages relating to files are generally reports of errors returned by Windows. You may find some of these messages (for example, "Access denied") vague enough that they are not always helpful. The command processor includes the file name in file error messages, but is often unable to determine a more accurate explanation of these errors. The message shown is the best information available based on the error codes returned by Windows.

Not all errors potentially reported by Windows can be listed here. See Windows System Errors 536 for examples of system errors returned in the _SYSERR 422 internal variable.

The following list includes most common error messages, in alphabetical order:

Access denied: You tried to write to or erase a read-only file, rename a file or directory to an existing name, create a directory that already exists, remove a read-only directory or a directory with files or subdirectories still in it, or access a file in use by another program in a multitasking system.

Alias loop: An alias refers back to itself either directly or indirectly (*i.e.*, a = b = a), or aliases are nested more than 16 deep. Correct your alias list.

Already debugging a batch file: You are attempting to invoke a nested instance of the Batch File Debugger (BDEBUGGER 201) while you are already in the debugger.

Already excluded files: You used more than one exclude range in a command. Combine the exclusions into a single range.

Bad disk unit: Generally caused by a disk drive hardware failure.

Batch file missing: the command processor can't find the batch (*.BTM* or *.CMD*) file it was running. It was either deleted, renamed, moved, or the disk was changed. Correct the problem and rerun the file.

Can't COPY or MOVE file to itself: You cannot COPY or MOVE a file to itself. The command processor attempts to perform full path and filename expansion before copying to help ensure that files aren't inadvertently destroyed.

Can't create: the command processor can't create the specified file. The disk may be full or write protected, or the file already exists and is read-only, or the root directory is full.

Can't delete: the command processor can't delete the specified file or directory. The disk is probably write protected.

Can't end current process: You attempted to terminate the command processor with a TASKEND 373 command. TASKEND can only be used to end other processes; to terminate the command processor, use the EXIT 262 command.

Can't get directory: the command processor can't read the directory. The disk drive is probably not ready.

Can't install hook: the command processor can't install the operating system hook required by the KEYSTACK 296 command. Contact JP Software for assistance.

Can't make directory entry: the command processor can't create the filename in the directory. This is usually caused by a full root directory. Create a subdirectory and move some of the files to it.

Can't open: the command processor can't open the specified file. Either the file doesn't exist or the disk directory or File Allocation Table is damaged.

Can't query key type: The key name supplied to @REGQUERY refers to a key with a type that @REGQUERY does not support. See @REGQUERY 485 for a list of supported key types.

Can't remove current directory: You attempted to remove the current directory, which Windows does not allow. Change to the parent directory and try again.

CD-ROM door open or **CD-ROM not ready**: The CD-ROM drive door is open, the power is off, or the drive is disconnected. Correct the problem and try again.

CD-ROM not High Sierra or ISO-9660: The CD-ROM is not recognized as a data CD (it may be a music CD). Put the correct CD in the drive and try again.

Clipboard is empty or not text format: You tried to retrieve some text from the Windows clipboard,

but there is no text available. Correct the contents of the clipboard and try again.

Clipboard is in use by another program: the command processor could not access the Windows clipboard because another program was using it. Wait until the clipboard is available, or complete any pending action in the other program, then try again.

Command line too long: A single command or the entire command line exceeded the maximum allowable length 71 (including during alias, variable, or function expansion). Reduce the complexity of the command or use a batch file. Also check for an alias which refers back to itself either directly or indirectly.

Command only valid in batch file: You have tried to use a batch file command, like DO or GOSUB, from the command line or in an alias. A few commands can only be used in batch files (see the individual commands for details).

Contents lost before copy: COPY was appending files, and found one of the source files is the same as the destination. That source file is skipped, and appending continues with the next file.

Data error: Windows can't read or write properly to the device. On a floppy drive, this error is usually caused by a defective floppy disk, dirty disk drive heads, or a misalignment between the heads on your drive and the drive on which the disk was created. On a hard drive, this error may indicate a drive that is too hot or too cold, or a hardware problem. Retry the operation; if it fails again, correct the hardware or diskette problem.

DDE [error message]: ***(TC)*** A DDE transaction could not be completed by the DDEEXEC 224 command. The error message explains the reason. Consult the documentation for the DDE server you are using for additional details if necessary.

Directory stack empty: POPD 326 or DIRS 245 can't find any entries in the directory stack.

Disk is write protected: The disk cannot be written to. Check the disk and remove the write-protect tab or close the write- protect window if necessary.

Divide by zero: The command or function you used tried to do a division by zero. If the data causing the problem is from your own input or batch file, change the input to avoid the divide by zero condition. If the data was generated internally by the command processor, contact JP Software for assistance.

Drive not ready — close door: The removable disk drive door is open. Close the door and try again.

Duplicate redirection: You tried to redirect standard input, standard output, or 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 3 command line, but the directive is in error. Usually a more specific error message follows, and can be looked up in this list.

Error on line [nnnn] of [filename]: There is an error in your .INI file 91. The following message explains the error in more detail. Correct the line in error and restart the command processor for your change to take effect.

Error reading: Windows experienced an I/O error when reading from a device. This is usually caused by a bad disk, a device not ready, or a hardware error.

Error writing: Windows experienced an I/O error when writing to a device. This is usually caused by a full disk, a bad disk, a device not ready, or a hardware error.

Exceeded batch nesting limit: You have attempted to nest batch files more than 10 levels deep.

File Allocation Table bad: Windows can't access the FAT on the specified disk. This can be caused by a bad disk, a hardware error, or an unusual software interaction.

File association not found: The ASSOC 197 command could not find a file association for the specified extension in the Windows registry.

File exists: The requested output file already exists, and the command processor won't overwrite it.

File not found: the command processor couldn't find the specified file. Check the spelling and path name.

File type not found: The FTYPE 274 command could not find the specified file type in the Windows registry.

General failure: This is usually a hardware problem, particularly a disk drive failure or a device not properly connected to a serial or parallel port. Try to correct the problem, or reboot and try again. See also: **Data error** above.

Include file not found: You used the Include directive in the .INI file 91, but the file you specified was not found or could not be opened.

Include files nested too deep: You used the Include directive in the .INI file 91, and attempted to nest include files more than three levels deep.

Infinite COPY or MOVE loop: You tried to COPY or MOVE a directory to one of its own subdirectories and used the /S switch, so the command would run forever. Correct the command and try again.

Input and output files must have different names: (BATCOMP 200) You are attempting to compress a file to itself.

Input file is already compressed: (BATCOMP 200) You are attempting to compress a batch file that has already been compressed.

Insufficient disk space: COPY or MOVE ran out of room on the destination drive. Remove some files and retry the operation.

Invalid batch file: The batch file is corrupted, or improperly compressed, 173 or encrypted. Retry with a new copy of the file.

Invalid character value: You gave an invalid value for a character directive in the .INI file 91.

Invalid choice value: You gave an invalid value for a "choice" directive (one that accepts a choice from a list, like "Yes" or "No") in the INI file 91.

Invalid color: You gave an invalid value for a color directive in the INI file 91.

Invalid count: The character repeat count for KEYSTACK 296 is incorrect.

Invalid date: An invalid date was entered. Check the syntax and reenter.

Invalid directive name: the command processor can't recognize the name of a directive in the INI file 91.

Invalid drive: A bad or non-existent disk drive was specified.

Invalid key name: You tried to make an invalid key substitution in the INI file 91, or you used an invalid key name in a keystroke alias 187 or command. Correct the error and retry the operation.

Invalid numeric value: You gave an invalid value for a numeric directive in the INI file 91.

Invalid parameter: the command processor didn't recognize a parameter. Check the syntax and spelling of the command you entered.

Invalid path: The specified path does not exist. Check the disk specification and/or spelling.

Invalid path or file name: You used an invalid path or filename in a directive in the .INI file 91.

Invalid time: An invalid time was entered. Check the syntax and reenter.

Keystroke substitution table full: the command processor ran out of room to store keystroke substitutions 102 entered in the .INI file 91. Reduce the number of key substitutions or contact JP Software or your dealer for assistance.

Label not found: A GOTO 282 or GOSUB 280 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 296 command is missing a closing parentheses around a character group. Correct the command.

Missing ENDTEXT: A TEXT 376 command is missing a matching ENDTEXT. Check the batch file.

Missing GOSUB: the command processor cannot perform the RETURN 340 command in a batch file. You tried to do a RETURN without a GOSUB 280, or your batch file has been corrupted.

Missing SETLOCAL: An ENDLOCAL 256 was used without a matching SETLOCAL 358.

No aliases defined: You tried to display aliases but no aliases have been defined.

No closing quote: the command processor couldn't find a second matching back quote [`] or double-quote ["] on the command line.

No expression: The expression passed to the %@EVAL 451 variable function is empty. Correct the expression and retry the operation.

No shared memory found: The SHRALIAS 361 command could not find any global alias list, history list, or directory history list to retain, because you executed the command from a session with local lists. Start the command processor with at least one global list, then invoke SHRALIAS.

No SMTP server: SENDMAIL 349 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 333 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 337. You cannot rename a file to a different disk drive.

Out of function space: You are attempting to create a User-defined Function 275 that would require more resources than what your system makes available. Shorten the function definition or delete functions you no longer need

Out of memory: the command processor or Windows had insufficient memory to execute the last command. Try to free some memory by closing other sessions. If the error persists, contact JP Software for assistance.

Out of paper: Windows detected an out-of-paper condition on one of the printers. Check your printer and add paper if necessary.

Overflow: An arithmetic overflow occurred in the @EVAL 451 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 361 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 361 command to unload SHRALIAS.EXE, but it was never loaded. This message is informational and may not indicate an error condition.

Startup failed, contact JP Software: the command processor could not initialize and start operation. Contact JP Software or your dealer for assistance.

String area overflow: the command processor ran out of room to store the text from string directives in the .INI file 91. Reduce the complexity of the *.INI* file or contact JP Software for assistance.

String too long: You tried to put more than 2038 characters into the KEYSTACK 296 buffer. Reduce the number of characters you are trying to send to the application at one time.

Syntax error: A command or variable function 424 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 451 variable function. Correct the expression and retry the operation.

UNKNOWN_CMD loop: The UNKNOWN_CMD alias 187 called itself more than ten times. The alias probably contains an unknown command itself, and is stuck in an infinite loop. Correct the alias.

Unknown command: A command was entered that the command processor didn't recognize and couldn't find in the current search path. Check the spelling or PATH specification. You can handle unknown commands with the UNKNOWN_CMD alias (see ALIAS 187).

Unknown option name: (OPTION 317) You are attempting to modify or display an invalid or unknown option name.

Unknown process: TASKEND 373 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 186 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.

10 Reference Information

10.1 CMD.EXE Comparison

THIS TOPIC IS STILL UNDER CONSTRUCTION. COMMENTS WELCOME.

The comparison of commands available is based on the version of **CMD.EXE** included with Windows XP Build 2600 Service Pack 2.

If the CMD.EXE command name matches an internal JP command, the JP command is either identical, or is enhanced.

4NT/TC command equivalent in CMD.EXE

4NT/TC command	***CMD.EXE command*** (nearest functionality)	***CMD.EXE command type***
? 410	HELP	external
ACTIVATE 186		
ALIAS 187	DOSKEY	external
ASSOC 197	ASSOC	internal
ATTRIB 198	ATTRIB	external
BATCOMP 200		
BDEBUGGER 201		
BEEP 208		
BREAK 209	BREAK	internal
BREAKPOINT 209		
CALL 209	CALL	internal
CANCEL 211		
CD 211	CD	internal
CDD 213		
CHCP 215	CHCP	external
CHDIR 211	CHDIR	internal
CLS 215	CLS	internal
COLOR 216	COLOR	internal
COPY 216	COPY	internal
COPY 216	XCOPY	external
DATE 224	DATE	internal
DEL 225	DEL	internal
DELAY 229		
DESCRIBE 230		
DETACH 232		
DIR 233	DIR	internal
DIRHISTORY 244		
DIRS 245		
DO 246		
DRAWBOX 250		
DRAWHLINE 251		
DRAWVLINE 252		
ECHO 253	ECHO	internal
ECHOERR 254		
ECHOS 255		
ECHOSERR 256		

ENDLOCAL 256	ENDLOCAL	internal
ERASE 225	ERASE	internal
ESET 258		
EVENTLOG 259		
EXCEPT 260		
EXIT 262	EXIT	internal
FFIND 262	FIND	external
FFIND 262	FINDSTR	external
FOR 267	FOR	internal
FREE 274		
FTYPE 274	FTYPE	internal
FUNCTION 275		
GLOBAL 279		
GOSUB 280	CALL :LABEL	internal
GOTO 282	GOTO	internal
HEAD 283		
HELP 284	HELP	external
HISTORY 285		
IF 286	IF	internal
IFF 287		
IFTP 288		
INKEY 291		
INPUT 293		
KEYBD 295		
KEYS 295		
KEYSTACK 296		
LIST 298	MORE	external
LOADBTM 303		
LOG 303		
MD 305	MD	internal
MEMORY 306		
MKDIR 305	MKDIR	internal
MKLNK 307		
MOVE 308	MOVE	internal
MSGBOX 313		
ON 315		
OPTION 317		
OSD 318		
PATH 319	PATH	internal
PAUSE 320	PAUSE	internal
PDIR 320		
PLAYAVI 324		
PLAYSOUND 325		
POPD 326	POPD	internal
PRINT 328	PRINT	external
PRIORITY 328		
PROMPT 329	PROMPT	internal
PUSHD 331	PUSHD	internal
QUERYBOX 332		
QUIT 333	GOTO :EOF	

RD 333	RD	internal
REBOOT 335		
RECYCLE 336		
REM 336	REM	internal
REN 337	REN	internal
RENAME 337	RENAME	internal
RETURN 340	GOTO :EOF	
REXEC 340		
RSHELL 341		
RMDIR 333	RMDIR	internal
SCREEN 342		
SCRPUT 344		
SELECT 345		
SENDMAIL 349		
SET 351	SET	internal
SETDOS 354		
SETLOCAL 358	SETLOCAL	internal
SHIFT 359	SHIFT	internal
SHORTCUT 360		
SHRALIAS 361		
SMPP 362		
SNPP 363		
START 364	START	internal
SWITCH 367		
SYNC 369		
TAIL 371		
TASKEND 373		
TASKLIST 374		
TEE 376		
TEXT 376		
TIME 378	TIME	internal
TIMER 379		
TITLE 380	TITLE	internal
TOUCH 381		
TREE 383	TREE	external
TRUENAME 384		
TYPE 384	TYPE	internal
UNALIAS 386		
UNFUNCTION 387		
UNSET 388	SET VARNAME=	internal
VER 389	VER	internal
VERIFY 390	VERIFY	internal
VOL 390	VOL	internal
VSCRPUT 391		
WHICH 391		
WINDOW 392		
Y 395		

CMD.EXE command equivalents in 4NT/TC

CMD.EXE command	*CMD.EXE class*	*4NT command*	*comparison*
ASSOC	internal	ASSOC 197	enhanced
AT	external		
ATTRIB	external	ATTRIB 198	enhanced
BREAK	internal	BREAK 209	enhanced
CACLS	external		
CALL	internal	CALL 209, GOSUB 280	enhanced
CD	internal	CD 211	enhanced
CHCP	external	CHCP 215	identical
CHDIR	internal	CHDIR 211	enhanced
CHKDSK	external		
CHKNTFS	external		
CLS	internal	CLS 215	enhanced
COLOR	internal	COLOR 216	enhanced
COMP	external		
COMPACT	external		
CONVERT	external		
COPY	internal	COPY 216	enhanced
DATE	internal	DATE 224	enhanced
DEL	internal	DEL 225	enhanced
DIR	internal	DIR 233, PDIR 320	enhanced
DISKCOMP	external		
DISKCOPY	external		
DOSKEY	external	ALIAS 187	enhanced
ECHO	internal	ECHO 253	identical
ENDLOCAL	internal	ENDLOCAL 256	enhanced
ERASE	internal	ERASE 225	enhanced
EXIT	internal	EXIT 262	enhanced
FC	external		
FIND	external	FFIND 262	enhanced
FINDSTR	external	FFIND 262	enhanced
FOR	internal	FOR 267	enhanced
FORMAT	external		
FTYPE	internal	FTYPE 274	enhanced
GOTO	internal	GOTO 282	enhanced
GRAFTABL	external		
HELP	external	HELP 284	enhanced
IF	internal	IF 286	enhanced
LABEL	external		
MD	internal	MD 305	enhanced
MKDIR	internal	MKDIR 305	enhanced
MODE	external		
MORE	external	LIST 298	enhanced
MOVE	internal	MOVE 308	enhanced
PATH	internal	PATH 319	identical
PAUSE	internal	PAUSE 320	enhanced
POPD	internal	POPD 326	enhanced
PRINT	external	PRINT 328	enhanced

PROMPT	internal	PROMPT 329	enhanced
PUSHD	internal	PUSHD 331	enhanced
RD	internal	RD 333	enhanced
RECOVER	external		
REM	internal	REM 336	identical
REN	internal	REN 337	enhanced
RENAME	internal	RENAME 337	enhanced
REPLACE	external		
RMDIR	internal	RMDIR 333	enhanced
SET	internal	SET 351	enhanced
SETLOCAL	internal	SETLOCAL 358	enhanced
SHIFT	internal	SHIFT 359	enhanced
SORT	external		
START	internal	START 364	enhanced
SUBST	external		
TIME	internal	TIME 378	enhanced
TITLE	internal	TITLE 380	enhanced
TREE	external	TREE 383	enhanced
TYPE	internal	TYPE 384	enhanced
VER	internal	VER 389	enhanced
VERIFY	internal	VERIFY 390	enhanced
VOL	internal	VOL 390	identical
XCOPY	external	COPY 216	enhanced

Command line editing

Filename completion

Command completion

Redirection

Wildcards

Built-In Variables

Batch File Structure

Unique Jpsoft command processor features

Batch debugger

Aliases

Internal functions

User defined functions

File selection

Ranges

Internet access and email

OpenAFS support

ANSI X3.64 support

Directory navigation

Histories and logs

Intersession sharing

Perl, REXX, and Ruby support

10.2 Limits

This topic is still under construction.

The categories of information below are expected to be collected and presented tabularly.

Length Limits

entity	name	value	all
environment variable	1023	8,191	8,191
alias	1023	16,380	16,383
user defined function	1023	16,380	16,383

command type	before expansion	after expansion
command line	8,191	16.383
command group	8,191	16,383

Nesting Limits

command	depth
CALL 209	32
DO 246	no limit
FOR 267	no limit
GOSUB 280 without parameters	no limit
GOSUB 280 with parameters	22
SETLOCAL 358	16
IFF 287	no limit

Miscellaneous Limits

entity	limit

character count in any function	16,383
number of batch file parameters	511
number of GOSUB parameters	255
file name	2,047
include list	2,047
single parameter	2,047
global alias list	131,072
global function list	65,536
key substitution table	128
directory stack	511

10.3 File Systems & File Name Conventions

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

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

The topics below are roughly divided according to the different parts of a file name, and cover the file system structure and naming conventions:

10.3.1 Drives & Volumes

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

Normally, drive **C:** is the first (or only) hard disk drive. Most current operating systems can partition a large hard disk into multiple logical drives or volumes that are usually called **C:**, **D:**, **E:**, etc. Network systems (LANs) give additional drive letters to sections of the network file server drives. In addition, you can access network drives via their **UNC** (universal naming convention) name (*e.g.* *data**vol1**...*), without using a drive letter. See File Systems 521 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 521 for details. Additional information about disk files and directories is available under Directories and Subdirectories 522, File Names 523, and File Attributes 524.

10.3.2 File Systems

Each disk volume is organized according to a file system. The file system determines how files are named, how they are organized on the disk, and what information about each file is stored in the file directory.

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

The operating systems under which the command processor runs support four standard file systems: FAT, VFAT, FAT32, and NTFS. Throughout this manual, the term "LFN file system" is commonly used to describe the VFAT, FAT32, and NTFS systems as a group, where. LFN stands for **L**ong **F**ile **N**ame. See File Names 523 for details on the rules for naming files under each file system.

Additional file systems may be installed under some operating systems to support CD, DVD and network drives.

The file system type is determined when a hard disk volume is formatted and applies to the entire volume. For example, you might have a 600 GB hard disk divided into three 149.5 GB volumes and a 500 MB volume, with the first three volumes (C:, D:, and E:) formatted for NTFS, and the fourth formatted for one of FAT/VFAT/FAT32.

The command processor supports any standard file system installed under your operating system, and can access all files supported by the operating system.

Additional information about disk files and directories is available under Drives and Volumes 520, Directories and Subdirectories 522, File Names 523, and File Attributes 524.

Network File Systems

A network file system allows you to access files stored on another computer on a network, rather than on your own system. The command processor supports all network file systems which are compatible with the underlying operating system. The networking software used to access remote systems (such as Unix, Linux, MacOS, etc..) which use different file systems typically emulates one of the common Windows file systems. Those emulations do not always provide a perfect duplicate of some functions (attributes, timestamps, etc.), an issue unrelated to the command processor.

File and directory names for network file systems depend on both the "server" software running on the system that has the files on it, and the "client" software running on your computer to connect it to the network. However, they usually follow the rules described here.

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

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

In rare cases, the command processor may not be able to report correct statistics on network drives (such as the number of bytes free on a drive). This is usually because the network file system does not provide complete or accurate information.

Universal Naming Convention (UNC)

Some networks also support the Universal Naming Convention, which provides a common method for accessing files on a network drive without using a mapped drive letter. Names specified this way are called UNC names. They typically appear as ***server**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 522.

The command processor also allows you to use UNC directory names when changing directories (see Directory Navigation 12 for more details).

OpenAFS

4NT and **TC** have built-in support for OpenAFS. The parser will recognize Unix-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.

10.3.3 Directories & Subdirectories

A file system is a method of organizing all of the files on an entire disk or hard disk volume. Directories (or folders) are used to divide the files on a disk into logical groups that are easy to work with. Their purpose is similar to that of file drawers containing groups of hanging folders, hanging folders containing smaller folders, and so on. (The terms directory and folder are not synonymous but often used as such in common Windows terminology. For accuracy, we use **directory** throughout these help files unless other folder types are also specifically applicable.)

Every drive has a root or base directory, and many have one or more subdirectories. Subdirectories can also have subdirectories, extending in a branching tree structure from the root directory. The collection of all directories on a drive is often called the directory tree, and a portion of the tree is sometimes called a subtree. The terms directory and subdirectory are typically used interchangeably to mean a single subdirectory within this tree structure.

Subdirectory names follow the same naming rules as files in each operating system (see File Names 523).

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 ***4NT*** and ***TC***, the path and filename can be up to 2047 characters, though many Windows applications (including **CMD.EXE** and **Explorer**) have trouble with path and filename lengths exceeding 260 characters. Shorter paths and names are advisable under Windows whenever feasible.

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

drive is specified) and the current directory (if no other directory path is specified).

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

There are also two special subdirectory names that are useful in many situations: a single period by itself [**.**] means "the current default directory." Two periods together [**..**] means "the directory which contains the current default directory" (often referred to as the parent directory). These special names can be used wherever a full directory name can be used. The command processor allows you to use additional periods to specify directories further "up" the tree (see Extended Parent Directory Names 62).

Additional information about disk files and file systems is available under Drives and Volumes 520, File Systems 521, File Names 523, and File Attributes 524.

10.3.4 File Names

FAT File Names

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

You can use alphabetic and numeric characters plus the punctuation marks **! # $ % & ' ( ) -** @ **^ _ ` { } ~** and in both the base name and the extension of a FAT filename. Because the exclamation point [**!**], percent sign [**%**], caret [**^**], at sign [@], parentheses [**()**], and back-quote [`] also have other meanings to the command processor, it is best to avoid using them in filenames. It is also better to use only those characters found in ASCII 541, because changing font and/or code page may change drastically how they are displayed.

FAT file names are always stored on the disk in upper case, and are displayed in upper or lower case depending on the options you select in the command processor.

Long File Names

VFAT, **FAT32** and **NTFS** allow using long file names with a maximum of 255 characters, including spaces and other characters that are not allowed in a FAT system file name, but excluding some punctuation characters which are allowed in FAT file names. See your operating system documentation for details on the characters allowed. If you use file names which contain semicolons [**;**], see Wildcards 19 for details on avoiding problems with interpretation of those file names under the command processor.

LFNs are stored and displayed exactly as you entered them, and are not automatically shifted to upper or lower case. For example, you could create a file called *MYFILE*, *myfile*, or *MyFile*, and each name would be stored in the directory just as you entered it. However, case is ignored when looking for filenames, so you cannot have two files whose names differ only in case (*i.e.*, the three names given above would all refer to the same file). This behavior is sometimes described as "case-retentive but not case-sensitive" because the case information is retained, but does not affect access to the files. This is in contrast with Unix-style file systems, which are case sensitive, and permit **AA**, **Aa**, **aA**, and **aa** to be four different file names.

A file that has an LFN may have an additional, "FAT-compatible" name, which contains only those characters legal on a FAT volume, and which meets the 8-character name / 3-character extension limits. Programs which cannot handle long names (for example, DOS programs accessing an NTFS drive generally can access files by using their FAT-compatible names. This name is assigned at the time the LFN is created in the specific directory, and to make it unique, it depends on what other SFNs exist in that directory at that instance. Consequently, when copying the file to another directory by its LFN the SFN generated in the target directory may be different from the SFN in the source directory.

When specifying an LFN-compatible file name, which includes spaces or other characters that would either not be allowed in a FAT name, or that may have syntactical significance for the command processor, you must place 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*.

You may occasionally encounter filenames which are not displayed the way you expect if you have used characters from outside the U.S. English character set in the name. These are generally due to problems in the way Windows translates characters between the OEM and ANSI character sets. Correcting the problem may require experimentation with fonts, character sets, and code pages, and occasionally some such problems may not be readily correctable within the command processor. For more information on underlying issues related to fonts and character sets see ASCII, Key Codes, and ANSI X3.64 Commands 540.

Additional information about disk files and file systems is available under Drives and Volumes 520, File Systems 521, Directories and Subdirectories 522, File Attributes 524, and Time Stamps 525.

10.3.5 File Attributes

Each file has attributes, each of which defines a single characteristic of the file that can be either set or reset. Most file processing commands allow you to select files for processing based on their attributes. The basic attributes Archive, Read only, Hidden, System, and Directory are present on all disk volumes. NTFS volumes support additional attributes: Encrypted, Compressed, Normal, Offline, Temporary, Not content-indexed, Sparse, and Junction / Reparse point. ***4NT*** and ***TC*** fully support these extended attributes 28.

Archive - set by the operating system when the contents of the file are modified to indicate that it is *a candidate to be archived*, i.e., to be backed up. The attribute can be reset by any program to indicate that the file's contents have been archived. Most programs which can unset this attribute require that you use the explicit reset option, and default to retaining the status of this attribute. For example, the command processor command **COPY** requires the ***/X*** option to reset this attribute.

Read-only – if this attribute is set, the file can't be changed or erased accidentally. Most programs honor this attribute by default, which helps to protect important files from erasure and damage.

Either of the **Hidden** and **System** attributes, when set, prevent the file from appearing in directory listings and file searches, including those performed by file processing command of the command processor, unless explicitly requested.. 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). JP Software command processors do not directly modify the volume label or any of its attributes, and provide read access only through the VOL 390 command and the @LABEL[] 476 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 28 available only on an NTFS volumes) are reset. It is not stored by the file system. When ***4NT*** or ***TC*** check file attributes, they consider the Normal attribute as set if each of the other attributes is either reset, or unsupported by the combination of the file system and operating system.

The file attributes can also be accessed with the ATTRIB 198 and DIR 233 commands, and by the @ATTRIB 439 and @WATTRIB 494 variable functions.

Attributes can be set, reset, and viewed with the ATTRIB 198 command. The DIR 233 command also has options to view the attribute status of files, and to view information about normally invisible hidden and system files and directories.

10.3.6 File Time Stamps

Each file has one or more time stamps. They are used by the operating system to record when the file was created, last modified, or last accessed. Most ***4NT*** and ***TC*** file processing commands allow you to select files for processing based on their time stamps.

1. **Write time** is the date and time the file was last written, i.e., when its content was last modified**,** On FAT volumes this is the only timestamp. In all commands and functions this is the timestamp used unless you specify another. On FAT and VFAT volumes, the resolution is 2 s. NTFS volumes have a 100 nanosecond resolution for the file creation and last write. (Unix and Linux systems use 1-s resolution.) When a file is copied using the COPY command, even across a network, its write time is not changed. However, different file systems record time with different resolution, so minor changes may occur.

2. **Creation time** is the date and time the current instance of the file was created.

3. **Access time** is the date, and on NTFS volumes, the time, when the file was last accessed for either reading or writing.

Several command processor commands and functions let you specify which set of time and date stamps you want to view or work with on LFN volumes. These commands and functions use the letter

c creation time stamp,
w last write time stamp, and
a last access time stamp.

Note that FAT32 and VFAT volumes store the date but not the time of the last access. On these drives the time of last access will always be 00:00.

Time Stamp Resolution

The resolution of time stamps as well as the range of time instances representable vary with file systems. The table below shows some of them.

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 00:00:01 *UTC*	
Unix/Linux	1 s	1970-01-01 00:00:00 *UTC*	

NTFS Timestamp Reports

These operating systems report timestamps in local time. However, conversion between UTC and local time is based on the difference between UTC and local time at the time of conversion, instead of that in effect when the file event occurred. Consequently, if daylight saving time is currently in effect, all file events around the year will be reported in DST. conversely, when DST is not in effect, all file events around the year will be reported in standard time. This method has the advantage that differences in event times can be calculated easily. However, the times reported will not be those when the event took place if the state DST at time of event is not the same as at the time of reporting.

The TOUCH 381 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 520, File Systems 521, Directories and Subdirectories 522, and File Names 523.

10.3.7 NTFS File Streams

The NTFS file system allows each file to contain multiple "streams" or sets of data. For example a compiler could use streams to store a program's source code, object code, and other data, or a word processing program could use them to store multiple versions of the same document.

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

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

You cannot use wildcards in stream names except when using filename completion 399.

You can display stream names with the DIR 233 ***/***: option. The file processing commands COPY 216, DEL 225, FFIND 262, LIST 298, MOVE 308 and TYPE 384 support file streams when the stream name is explicitly specified; see the individual commands for additional details. Other file-related commands, such as ATTRIB, REN and TOUCH work with the file as a whole, and not with any particular stream or portion of the file data.

Variable functions which reference file contents, such as @FILEOPEN 459, @LINE 477, and @LINES 478 also accept stream names.

DIR 233 and other commands and functions which return file information (TREE 383, @FILESIZE 462, etc.) reference only the normal file data, and do not include stream data.

10.4 Regular Expression Syntax

Oniguruma Regular Expressions Version 4.2.0 2006/07/18

This section covers the Ruby regular expression syntax. For information on Perl regular expression syntax, see your Perl documentation or http://www.perl.com/doc/manual/html/pod/perlre.html.

1. Syntax elements

```
\       escape (enable or disable meta character meaning)
|       alternation
(...)   group
[...]   character class
```

2. Characters

```
\t          horizontal tab (0x09)
\v          vertical tab   (0x0B)
\n          newline        (0x0A)
\r          return         (0x0D)
\b          back space     (0x08)
\f          form feed      (0x0C)
\a          bell           (0x07)
\e          escape         (0x1B)
\nnn        octal char           (encoded byte value)
\xHH        hexadecimal char     (encoded byte value)
\x{7HHHHHHH} wide hexadecimal char (character code point value)
\cx         control char         (character code point value)
\C-x        control char         (character code point value)
\M-x        meta  (x|0x80)       (character code point value)
\M-\C-x     meta control char    (character code point value)

(* \b is effective in character class [...] only)
```

3. Character types

```
.       any character (except newline)

\w      word character

        Not Unicode:
                alphanumeric, "_" and multibyte char.

        Unicode:
            General_Category -- (Letter|Mark|Number|Connector_Punctuation)

\W      non word char

\s      whitespace char

        Not Unicode:
            \t, \n, \v, \f, \r, \x20

        Unicode:
            0009, 000A, 000B, 000C, 000D, 0085(NEL),
            General_Category -- Line_Separator
                -- Paragraph_Separator
                 -- Space_Separator
```

\S non whitespace char

\d decimal digit char

Unicode: General_Category -- Decimal_Number

\D non decimal digit char

\h hexadecimal digit char [0-9a-fA-F]

\H non hexadecimal digit char

4. Quantifier

greedy

? 1 or 0 times
* 0 or more times
+ 1 or more times
{n,m} at least n but not more than m times
{n,} at least n times
{,n} at least 0 but not more than n times ({0,n})
{n} n times

reluctant

?? 1 or 0 times
*? 0 or more times
+? 1 or more times
{n,m}? at least n but not more than m times
{n,}? at least n times
{,n}? at least 0 but not more than n times (== {0,n}?)

possessive (greedy and does not backtrack after repeated)

?+ 1 or 0 times
*+ 0 or more times
++ 1 or more times

({n,m}+, {n,}+, {n}+ are possessive op. in ONIG_SYNTAX_JAVA only)

ex. /a*+/ === /(?>a*)/

5. Anchors

^ beginning of the line
$ end of the line
\b word boundary
\B not word boundary
\A beginning of string
\Z end of string, or before newline at the end
\z end of string
\G matching start position (*)

6. Character class

^...	negative class (lowest precedence operator)
x-y	range from x to y
[...]	set (character class in character class)
..&&..	intersection (low precedence at the next of ^)

ex. [a-w&&[^c-g]z] ==> ([a-w] AND ([^c-g] OR z)) ==> [abh-w]

* If you want to use '[', '-', ']' as a normal character in a character class, you should escape these characters by '\'.

POSIX bracket ([: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	
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
xdigit	0030 - 0039 | 0041 - 0046 | 0061 - 0066 (0-9, a-f, A-F)

7. Extended groups

(?#...)	comment
(?imx-imx)	option on/off i: ignore case

	m: multi-line (dot(.) match newline) x: extended form
(?imx-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.

(?>subexp)	atomic group don't backtrack in subexp.
(?<name>subexp)	define named group (All characters of the name must be a word character. And first character must not be a digit or upper case) Not only a name but a number is assigned like a captured group.

Assigning the same name as two or more subexps is allowed. In this case, a subexp call can not be performed although the back reference is possible.

8. Back reference

\n	back reference by group number (n >= 1)
\k<name>	back reference by group name

In the back reference by the multiplex definition name, a subexp with a large number is referred to preferentially. (When not matched, a group of the small number is referred to.)

* Back reference by group number is forbidden if named group is defined in the pattern and ONIG_OPTION_CAPTURE_GROUP is not set.

Back reference with nest level

\k<name+n>	n: 0, 1, 2, ...
\k<name-n>	n: 0, 1, 2, ...

Destinate relative nest level from back reference position.

ex 1.

```
/\A(?<a>|.|(?:(?<b>.)\g<a>\k<b+0>))\z/.match("reer")
```

ex 2.

```
r = Regexp.compile(<<'__REGEXP__'.strip, Regexp::EXTENDED)
```

```
(?<element> \g<stag> \g<content>* \g<etag> ){0}
(?<stag> < \g<name> \s* > ){0}
(?<name> [a-zA-Z_:]+ ){0}
(?<content> [^<&]+ (\g<element> | [^<&]+)* ){0}
(?<etag> </ \k<name+1> >){0}
\g<element>
__REGEXP__

p r.match('<foo>f<bar>bbb</bar>f</foo>').captures
```

9. Subexp call ("Tanaka Akira special")

```
\g<name>   call by group name
\g<n>      call by group number (n >= 1)
```

* left-most recursive call is not allowed.
ex. (?<name>a|\g<name>b) => error
(?<name>a|b\g<name>c) => OK

* Call by group number is forbidden if named group is defined in the pattern and
ONIG_OPTION_CAPTURE_GROUP is not set.

* If the option status of called group is different from calling position then the group's option is effective.

ex. (?-i:\g<name>)(?i:(?<name>a)){0} match to "A"

10. Captured group

Behavior of the no-named group (...) changes with the following conditions. (But named group is not changed.)

case 1. /.../ (named group is not used, no option)

(...) is treated as a captured group.

case 2. /.../g (named group is not used, 'g' option)

(...) is treated as a no-captured group (?:...).

case 3. /..(?<name>..)../ (named group is used, no option)

(...) is treated as a no-captured group (?:...).
numbered-backref/call is not allowed.

case 4. /..(?<name>..)../G (named group is used, 'G' option)

(...) is treated as a captured group.
numbered-backref/call is allowed.

where
g: ONIG_OPTION_DONT_CAPTURE_GROUP
G: ONIG_OPTION_CAPTURE_GROUP

('g' and 'G' options are argued in ruby-dev ML)

These options are not implemented in Ruby level.

A-1. Syntax dependent options

+ ONIG_SYNTAX_RUBY
  (?m): dot(.) match newline

+ ONIG_SYNTAX_PERL and ONIG_SYNTAX_JAVA
  (?s): dot(.) match newline
  (?m): ^ match after newline, $ match before newline

A-2. Original extensions

+ hexadecimal digit char type \h, \H
+ named group (?<name>...)
+ named backref \k<name>
+ subexp call \g<name>, \g<group-num>

A-3. Lacked features compare with perl 5.8.0

+ [:word:]
+ \N{name}
+ \l,\u,\L,\U, \X, \C
+ (?{code})
+ (??{code})
+ (?(condition)yes-pat|no-pat)

* \Q...\E
  This is effective on ONIG_SYNTAX_PERL and ONIG_SYNTAX_JAVA.

* \p{property}, \P{property}
  This is effective on ONIG_SYNTAX_PERL and ONIG_SYNTAX_JAVA.
  Alnum, Alpha, Blank, Cntrl, Digit, Graph, Lower, Print, Punct, Space, Upper, XDigit, ASCII are supported.

  Prefix 'Is' of property name is allowed in ONIG_SYNTAX_PERL only.
  ex. \p{IsXDigit}.

  Negation operator of property is supported in ONIG_SYNTAX_PERL only.
  \p{^...}, \P{^...}

10.5 Miscellaneous Reference Information

- Executable Files and File Searches 533
- Primary and Secondary Shells 535
- Colors and Color Names 552
- Keys and Key Names 550
- Popup Windows 536
- Windows System Errors 536

10.5.1 Executable Files & File Searches

When the command processor can't find a matching internal command name, it tries to find an executable file whose name matches the command name. (Executable files are typically those with a *.COM or .EXE* extension.)

If the command processor cannot find an executable program to run, it next looks for a matching batch file 162 name. The command processor looks first for a *.BTM* file, then for a *.CMD* file, then for a *.BAT* file, and finally for a *.REX*, *.PL*, or *.RB* file (if REXX, Perl, and/or Ruby are enabled).

You can change the list of extensions that are considered "executable", and the order in which they are searched, with the PATHEXT 399 environment variable, and the related PathExt 135 directive in the .INI file 91. PATHEXT is supported for compatibility reasons but should not generally be used as a substitute for executable extensions 33, which are more flexible.

Note: If the search for an external program or batch file fails, the command processor checks to see if the command name matches the name of a file with an executable extension 33. If an executable extension is found, the command processor runs the program specified when the association was defined. If no executable extension is found, the command processor 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, the command processor calls the ShellExec Windows API. The command processor has no control over which action the above Windows APIs will take when presented with a file name. If you are concerned about what Windows might do with an "unknown" extension, create a specific executable extension.

The command processor first performs this search (for an executable program, a batch file, or a file with an executable extension) in the current directory. If that search fails, they repeat the search in every directory in your search path.

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

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

The directory names in the search path are separated by semicolons.

You can create or view the search path with the PATH 319 command. You can use the ESET 258 command to edit the path. Many programs also use the search path to find their own files. The search path is stored in the environment with the name PATH.

(TC) Take Command also searches the *\WINDOWS\SYSTEM32* directory followed by the *\WINDOWS* directory. (The actual directory names may be different on your system. The command processor will determine the correct names for the "Windows" and "Windows System" directories and use them.) This part of the search procedure conforms with the traditional search sequences used under each Windows operating system.

Note: If the file is not found on the PATH, the command processor then checks for a corresponding **App Paths** entry in the Windows registry. **App Paths** entries are created by some applications during the installation process.

Remember, the command processor always looks for an executable file (or a file with an executable extension or Windows file association) in the current subdirectory, then in the Windows directories if appropriate (see above), then in each directory in the search path, and then in the **App Paths** area of

the registry. (You can change the search order so the current directory is not searched first; see the PATH 319 command for details.)

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

If your command name includes a path, the elements must be separated with backslashes (*e.g.* **c:\wp\wp**). If you are accustomed to Unix syntax where forward slashes are used in command paths, and want the command processor to recognize this approach, you can set UnixPaths 148 to Yes in the .INI File 91.

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

If you specify a file name including extension, and the file exists in the current directory (or you specify a path), but the file does not have an extension known to the command processor (*.EXE, .COM, .BTM, .BAT, .CMD*, or an executable extension), then the file name will be passed to Windows to check for file associations defined in the Windows registry. This allows you to execute any file whose extension is known to Windows, simply by typing its name. For example, if you have no executable extension defined for *.PSP* files, but this is an extension known to Windows, at the prompt you can simply enter a command like this:

```
[c:\graphics] image1.psp
```

and the command processor will request that Windows start the application for you. See Windows File Associations 535 for additional details on how to control Windows file associations in the command processor.

The following table sums up the possible search options (the term "standard search" refers to the search of the current directory, the Windows directories in ***TC***, and each directory in the search path):

Command	Command Processor 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 command processor cannot find an executable file, batch program, or a file with an executable extension or Windows file association in the current directory, a directory in the search path, or the directory you specified in the command, it then looks for an alias called **UNKNOWN_CMD** (see the ALIAS 187 command for details). If you have defined an alias with that name, it is executed (this allows you to control error handling for unknown commands). Otherwise, the command processor

displays an "Unknown command" error message and waits for your next instruction.

See also: the WHICH 391 command.

10.5.1.1 Windows File Associations

Windows includes the ability to associate file extensions with specific applications; this feature is sometimes called "file associations". For example, 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, the command processor first searches for an external program file with a standard extension (*.COM*, *.EXE*, etc.). It then checks executable extensions. If all of these tests fail, the command processor passes the command name to Windows to see if Windows can find an association for it.

The command processor offers two commands which provide control over file associations. Both should be used with caution to avoid creating errors in the registry or damaging existing file types. The ASSOC 197 command modifies or displays the associations between extensions and file types in the Windows registry. The FTYPE 274 command modifies or displays the default command used to "open" a file of a specified type.

Executable extensions defined in the command processor always take precedence over file associations defined in Windows. For example, if you associate the *.TXT* extension with your own editor using a command processor executable extension, and Windows has associated *.TXT* with Notepad, your setting will have priority, and the association with Notepad will be ignored when you invoke a *.TXT* file from within the command processor.

See also:: START 364, ASSOC 197, FTYPE 274, Executable Extensions 33, Executable Files and File Searches 533.

10.5.2 Primary & Secondary Shells

The command processor is commonly referred to as the **shell.** These help files often make reference to **primary shell** and **secondary shell**.

A primary shell is an instance of the command processor which is not the child of another instance of the same command processor. It is the "first" copy started by some other process. A common type of primary shell would be an instance of ***4NT*** or ***TC*** launched by invoking (clicking on) a Windows shortcut. Note that a copy of ***TC*** launched from a ***4NT*** prompt would also be a primary shell since its parent will be "some other process", not a previous ***TC*** instance.

A secondary shell is an instance of the command processor launched from a previous instance of that same command processor. For example, if you are at a ***4NT*** prompt and invoke "d:\path\4nt.exe", then that new copy will be a secondary shell. In general, a secondary shell inherits most settings from the parent primary shell. A secondary shell can in turn launch additional secondary shells, but only the original instance is a primary shell.

In Windows, there is no "default command processor" and therefore no global primary shell. Each instance of ***4NT*** or ***TC*** invoked by clicking on a shortcut or entering its name in a "run" dialog is a primary shell. To facilitate communication between those several "parallel" instances of the command processor, we provide the ability to share resources via the LocalAliases 130, LocalHistory 131, LocalDirHistory 130 and LocalFunctions 131 directives and the SHRALIAS 361 command. Any ***4NT*** or ***TC*** instance launched from an existing copy ***4NT*** or ***TC*** will be a secondary shell. Note that secondary shells can be invoked explicitly (e.g. "d:\path\4nt.exe" from a ***4NT*** prompt) or implicitly (to execute a pipe, for example).

10.5.3 Popup Windows

Several features of the command processor display popup windows. A popup window may be used to display filenames, recently-executed commands, recently-used directories, the results of an extended directory search 14, or a list created by the SELECT 345 command or the @SELECT 489 internal function.

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

Navigation inside any popup window follows the conventions described below. Additional information on each specific type of popup window is provided where that window is discussed in detail.

You can control the position and size of most popup windows with the configuration dialogs 151, or with the PopupWinLeft 137, PopupWinTop 137, PopupWinWidth 137, and PopupWinHeight 137 directives in the .INI file 91. A few popup windows (e.g., the extended directory search window) have their own specific *.INI* directives, and corresponding separate choices in the configuration dialogs. You can also change the keys used in most popup windows with key mapping directives 102 in the .INI file.

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

Up Arrow 149	Move the selection bar up one line.
Down Arrow 116	Move the selection bar down one line.
Left Arrow 125	Scroll the display left 1 column, if it is a scrolling display (i.e. if it has a horizontal scrollbar).
Right Arrow 140	Scroll the display right 1 column, if it is a scrolling display (i.e. if it has a horizontal scrollbar).
PgUp 142	Scroll the display up one page.
PgDn 142	Scroll the display down one page.
Ctrl-PgUp 136 or Home 136	Go to the beginning of the list.
Ctrl-PgDn 137 or End 137	Go to the end of the list.
Esc 117	Close the window without making a selection.
Enter 137	Select the current item and close the window.

Note: The keystrokes shown above are the defaults values. See Key Mapping Directives 102 for details on how to assign different keystrokes.

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

10.5.4 Windows System Errors

This list shows a typical set of error and system messages returned by Windows. The numbers are what might be returned in internal variable _SYSERR 422. and the corresponding text is that provided by Microsoft.

Code	Error Message
0	The operation completed successfully.
1	Incorrect function.
2	The system cannot find the file specified.

3 The system cannot find the path specified.
4 The system cannot open the file.
5 Access is denied.
6 The handle is invalid.
7 The storage control blocks were destroyed.
8 Not enough storage is available to process this command.
9 The storage control block address is invalid.
10 The environment is incorrect.
11 An attempt was made to load a program with an incorrect format.
12 The access code is invalid.
13 The data is invalid.
14 Not enough storage is available to complete this operation.
15 The system cannot find the drive specified.
16 The directory cannot be removed.
17 The system cannot move the file to a different disk drive.
18 There are no more files.
19 The media is write protected.
20 The system cannot find the device specified.
21 The device is not ready.
22 The device does not recognize the command.
23 Data error (cyclic redundancy check).
24 The program issued a command but the command length is incorrect.
25 The drive cannot locate a specific area or track on the disk.
26 The specified disk or diskette cannot be accessed.
27 The drive cannot find the sector requested.
28 The printer is out of paper.
29 The system cannot write to the specified device.
30 The system cannot read from the specified device.
31 A device attached to the system is not functioning.
32 The process cannot access the file because it is being used by another process.
33 The process cannot access the file because another process has locked a portion of the file.
36 Too many files opened for sharing.
38 Reached the end of the file.
39 The disk is full.
50 The network request is not supported.
51 The remote computer is not available.
52 A duplicate name exists on the network.
53 The network path was not found.
54 The network is busy.
55 The specified network resource or device is no longer available.
56 The network BIOS command limit has been reached.
57 A network adapter hardware error occurred.
58 The specified server cannot perform the requested operation.
59 An unexpected network error occurred.
60 The remote adapter is not compatible.
61 The printer queue is full.
62 Space to store the file waiting to be printed is not available on the server.
63 Your file waiting to be printed was deleted.
64 The specified network name is no longer available.
65 Network access is denied.
66 The network resource type is not correct.
67 The network name cannot be found.
68 The name limit for the local computer network adapter card was exceeded.
69 The network BIOS session limit was exceeded.
70 The remote server has been paused or is in the process of being started.
71 No more connections can be made to this remote computer at this time because there are

already as many connections as the computer can accept.
72 The specified printer or disk device has been paused.
80 The file exists.
82 The directory or file cannot be created.
83 Fail on INT 24.
84 Storage to process this request is not available.
85 The local device name is already in use.
86 The specified network password is not correct.
87 The parameter is incorrect.
88 A write fault occurred on the network.
89 The system cannot start another process at this time.
100 Cannot create another system semaphore.
101 The exclusive semaphore is owned by another process.
102 The semaphore is set and cannot be closed.
103 The semaphore cannot be set again.
104 Cannot request exclusive semaphores at interrupt time.
105 The previous ownership of this semaphore has ended.
107 The program stopped because an alternate diskette was not inserted.
108 The disk is in use or locked by another process.
109 The pipe has been ended.
110 The system cannot open the device or file specified.
111 The file name is too long.
112 There is not enough space on the disk.
113 No more internal file identifiers available.
114 The target internal file identifier is incorrect.
117 The IOCTL call made by the application program is not correct.
118 The verify-on-write switch parameter value is not correct.
119 The system does not support the command requested.
120 This function is not supported on this system.
121 The semaphore timeout period has expired.
122 The data area passed to a system call is too small.
123 The filename, directory name, or volume label syntax is incorrect.
124 The system call level is not correct.
125 The disk has no volume label.
126 The specified module could not be found.
127 The specified procedure could not be found.
128 There are no child processes to wait for.
130 Attempt to use a file handle to an open disk partition for an operation other than raw disk I/O.
131 An attempt was made to move the file pointer before the beginning of the file.
132 The file pointer cannot be set on the specified device or file.
133 A JOIN or SUBST command cannot be used for a drive that contains previously joined drives.
134 An attempt was made to use a JOIN or SUBST command on a drive that has already been joined.
135 An attempt was made to use a JOIN or SUBST command on a drive that has already been substituted.
136 The system tried to delete the JOIN of a drive that is not joined.
137 The system tried to delete the substitution of a drive that is not substituted.
138 The system tried to join a drive to a directory on a joined drive.
139 The system tried to substitute a drive to a directory on a substituted drive.
140 The system tried to join a drive to a directory on a substituted drive.
141 The system tried to SUBST a drive to a directory on a joined drive.
142 The system cannot perform a JOIN or SUBST at this time.
143 The system cannot join or substitute a drive to or for a directory on the same drive.
144 The directory is not a subdirectory of the root directory.
145 The directory is not empty.
146 The path specified is being used in a substitute.

147 Not enough resources are available to process this command.
148 The path specified cannot be used at this time.
149 An attempt was made to join or substitute a drive for which a directory on the drive is the target of a previous substitute.
150 System trace information was not specified in your CONFIG.SYS file, or tracing is disallowed.
151 The number of specified semaphore events for DosMuxSemWait is not correct.
152 DosMuxSemWait did not execute; too many semaphores are already set.
153 The DosMuxSemWait list is not correct.
154 The volume label you entered exceeds the label character limit of the target file system.
155 Cannot create another thread.
156 The recipient process has refused the signal.
157 The segment is already discarded and cannot be locked.
158 The segment is already unlocked.
159 The address for the thread ID is not correct.
160 The argument string passed to DosExecPgm is not correct.
161 The specified path is invalid.
162 A signal is already pending.
164 No more threads can be created in the system.
167 Unable to lock a region of a file.
170 The requested resource is in use.
173 A lock request was not outstanding for the supplied cancel region.
174 The file system does not support atomic changes to the lock type.
180 The system detected a segment number that was not correct.
183 Cannot create a file when that file already exists.
186 The flag passed is not correct.
187 The specified system semaphore name was not found.
196 The operating system cannot run this application program.
197 The operating system is not presently configured to run this application.
199 The operating system cannot run this application program.
200 The code segment cannot be greater than or equal to 64K.
203 The system could not find the environment option that was entered.
205 No process in the command subtree has a signal handler.
206 The filename or extension is too long.
207 The ring 2 stack is in use.
208 The global filename characters, * or ?, are entered incorrectly or too many global filename characters are specified.
209 The signal being posted is not correct.
210 The signal handler cannot be set.
212 The segment is locked and cannot be reallocated.
214 Too many dynamic-link modules are attached to this program or dynamic-link module.
215 Cannot nest calls to LoadModule.
230 The pipe state is invalid.
231 All pipe instances are busy.
232 The pipe is being closed.
233 No process is on the other end of the pipe.
234 More data is available.
240 The session was canceled.
254 The specified extended attribute name was invalid.
255 The extended attributes are inconsistent.

CAUTION: This is a **large** topic (*>650 KB*). The remainder of this list has been omitted in the PDF version of this file. Consider viewing the copy in your local help file (*jphelp.chm*) instead!

d.

10.6 ASCII, Key Codes and Key Names

For ASCII codes, key codes and names see:

- ASCII Table 541
- Key Codes and Scan Codes Explanation 544
- Key Codes and Scan Codes Table 545
- Keys & Key Names 550

The remainder of this section gives a detailed explanation of character sets, ASCII, and key codes. For more information on ***4NT***'s and ***TC***'s ANSI X3.64 string support see ANSI X3.64 Commands Reference 550. 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.

For an explanation of how key codes work, see the Key Codes and Scan Codes Explanation 544. For a list of key codes and scan codes for keys on the standard U.S. keyboard see the Key Codes and Scan Codes Table 545.

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

The control codes can be entered on most keyboards by pressing the **Ctrl** key plus another character, or by pressing the special keys **Tab**, **Enter**, **Backspace**, and **Esc**.

See your operating system documentation for more information about character sets, code pages, and country and language support. Refer to your operating system and/or font documentation for details on the full character set available in any particular font.

The tables in this section are based on U.S. English conventions. Your system may differ if it is configured for a different country or language. See your operating system documentation for more information about country and language support.

Note: You may also be able to use the **Alt + keypad** approach to generate ASCII values. See "

Command Line Editing 47" for additional information.

10.6.1 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 540.

Control Characters 0 - 31, 127

ASCII (Dec)	ASCII (Hex)	Ctrl + Key	Acronym	Name
0	00	@	NUL	null
1	01	A	SOH	start of header
2	02	B	STX	start text
3	03	C	ETX	end text
4	04	D	EOT	end of transmission
5	05	E	ENQ	enquiry
6	06	F	ACK	acknowledge
7	07	G	BEL	bell
8	08	H	BS	backspace
9	09	I	HT	horizontal tab
10	0A	J	LF	linefeed
11	0B	K	VT	vertical tab
12	0C	L	FF	form feed
13	0D	M	CR	carriage return
14	0E	N	SO	shift out
15	0F	O	SI	shift in
16	10	P	DLE	data link escape
17	11	Q	DC1	device control 1
18	12	R	DC2	device control 2
19	13	S	DC3	device control 3
20	14	T	DC4	device control 4
21	15	U	NAK	negative acknowledge
21	16	V	SYN	synchronize

23	17	W	ETB	end text block
24	18	X	CAN	cancel
25	19	Y	EM	end of medium
26	1A	Z	SUB	substitute
27	1B	[	ESC	escape
28	1C	\	FS	field separator
29	1D	]	GR	group separator
30	1E	^	RS	record separator
31	1F	_	US	unit separator
127	7F	***n/a***	DEL	delete

Printing Characters 32 - 47

Dec	Hex	Char	Special character name
032	20	Space	space
033	21	!	exclamation mark
034	22	"	quote mark
035	23	#	number sign
036	24	$	dollar (currency) sign
037	25	%	percent mark
038	26	&	ampersand
039	27	'	apostrophe
040	28	(	left parenthesis
041	29	)	right parenthesis
042	2A	*	asterisk
043	2B	+	plus sign
044	2C	,	comma
045	2D	-	hyphen (minus sign)
046	2E	.	period
047	2F	/	slash

Printing Characters 48 - 63

Dec	Hex	Char	Special character name
048	30	0	
049	31	1	
050	32	2	
051	33	3	
052	34	4	
053	35	5	
054	36	6	
055	37	7	
056	38	8	
057	39	9	
058	3A	:	colon
059	3B	;	semicolon
060	3C	<	less than sign
061	3D	=	equal sign

062	3E	>	greater than sign
063	3F	?	question mark

Printing Characters 64 - 79

Dec	Hex	Char	Special character name
064	40	@	at sign
065	41	A	
066	42	B	
067	43	C	
068	44	D	
069	45	E	
070	46	F	
071	47	G	
072	48	H	
073	49	I	
074	4A	J	
075	4B	K	
076	4C	L	
077	4D	M	
078	4E	N	
079	4F	O	

Printing Characters 80 - 95

Dec	Hex	Char	Special character name
080	50	P	
081	51	Q	
082	52	R	
083	53	S	
084	54	T	
085	55	U	
086	56	V	
087	57	W	
088	58	X	
089	59	Y	
090	5A	Z	
091	5B	[	left bracket
092	5C	\	backslash
093	5D	]	right bracket
094	5E	^	caret
095	5F	_	underscore

Printing Characters 96 - 111

Dec	Hex	Char	Special character name
096	60	`	accent grave (back tick or back quote)
097	61	a	

098	62	b	
099	63	c	
100	64	d	
101	65	e	
102	66	f	
103	67	g	
104	68	h	
105	69	i	
106	6A	j	
106	6B	k	
108	6C	l	
109	6D	m	
110	6E	n	
111	6F	o	

Printing Characters 112 - 126

Dec	Hex	Char	Special character name
112	70	p	
113	71	q	
114	72	r	
115	73	s	
116	74	t	
117	75	u	
118	76	v	
119	77	w	
120	78	x	
121	79	y	
122	7A	z	
123	7B	{	left brace
124	7C	|	vertical bar
125	7D	}	right brace
126	7E	~	tilde

10.6.2 Key Codes and Scan Codes Explanation

This section explains how key codes and scan codes work. For a reference chart, see the Key Codes and Scan Codes Table 545. For more details on key codes, scan codes, and ASCII see the general information on ASCII, Key Codes, and ANSI X3.64 Commands 540.

When you press a single key or a key combination, Windows translates your keystroke into two numbers: a scan code, representing the actual key that was pressed, and an ASCII code, representing the ASCII value for that key. Windows returns these numbers the next time a program requests keyboard input. This section explains how key codes work; for information on using them with the command processor see the key mapping directives 102 in the .INI file 91, keystroke aliases 187, and INKEY 291.

Most command processor commands that use the numeric key codes listed here also use key names, which are usually more convenient to use than the numeric codes. See Keys and Key Names 550 for more information on key names.

As PCs have evolved, the structure of keyboard codes has evolved somewhat haphazardly with them, resulting in a bewildering array of possible key codes. We'll give you a basic explanation of how key codes work. For a more in-depth discussion, refer to a BIOS or PC hardware reference manual.

The nuances of how your keyboard behaves depends on the keyboard manufacturer, the computer manufacturer who provides the built-in BIOS, and your operating system. As a result, we can't guarantee the accuracy of the information in the tables for every system, but the discussion and reference table should be accurate for most systems. Our discussion is based on the "enhanced" keyboard commonly used on computers running Windows.

All keys have a scan code, but not all have an ASCII code. For example, function keys and cursor keys are not part of the ASCII character set and have no ASCII, but they do have a scan code. Some keys have more than one ASCII, depending on what other keys they are combined with. The "**A**" key, for example, has ASCII 97 (lower case "a") if you press it by itself (with "caps lock" off). If you press it along with **Shift** (with "caps lock" off), the code changes to 65 (upper case "A"). If you press **Ctrl** and **A,** the code changes to 1. In all these cases, the scan code (30) is unchanged because you are pressing the same physical key.

Things are different if you press **Alt-A**. **Alt** keystrokes have no ASCII, so Windows returns an ASCII of 0, along with the **A** key's scan code of 30. This allows a program to detect all the possible variations of **A**, based on the combination of ASCII and scan code.

Some keys generate more than one scan code depending on whether **Shift**, **Ctrl**, or **Alt** is pressed. This allows a program to differentiate between two different keystrokes on the same key, neither of which has a corresponding ASCII. For example, **F1** has no ASCII, so it returns code 0, and the **F1** scan code of 59. **Shift-F1** also returns a code 0; if it also returned a scan code of 59, a program couldn't distinguish it from **F1**. The operating system translates scan codes for keys like **Shift-F1** (and **Ctrl-F1** and **Alt-F1**) so that each variation returns a different scan code along with an ASCII 0.

On the enhanced keyboard there's one more variation: non-ASCII keys on the cursor keypad (such as up-arrow) return the same scan code as the corresponding key on the numeric keypad, for compatibility reasons. If they also returned an ASCII code of 0, a program couldn't tell which key was pressed. Therefore, these keys return an ASCII code of 224 rather than 0. This means that older programs, which only look for an ASCII 0 to indicate a non-ASCII keystroke like up-arrow, may not detect these cursor pad keys properly.

The number of different codes returned by any given key varies from one (for the space bar) to four, depending on the key, the design of your keyboard, and the operating system. Some keys, like **Alt**, **Ctrl**, and **Shift** by themselves or in combination with each other, plus **Print Screen**, **SysReq**, **Scroll Lock**, **Pause**, **Break**, **Num Lock**, and **Caps Lock** keys, do not have any code representations at all. The same is true of keystrokes with more than one modifying key, like **Ctrl-Shift-A**. The operating system may perform special actions automatically when you press these keys (for example, it switches into Caps Lock mode when you press **Caps Lock**), but it does not report the keystrokes to whatever program is running. Programs which detect such keystrokes access the keyboard hardware directly, a subject which is beyond the scope of this documentation.

10.6.3 Key Codes and Scan Codes Table

For more details on key codes, scan codes, and ASCII see the general information on ASCII, Key Codes, and ANSI X3.64 Commands 540, and the Key Codes and Scan Codes Explanation 544.

The table below shows standard codes used on a U.S. keyboard. Other keyboards will vary, due to the changes required to support characters outside the U.S. English character set.

Key names prefaced by **np** are on the numeric keypad. Those prefaced by **cp** are on the cursor keypad between the main typing keys and the number keypad. The numeric keypad values are valid if Num Lock is turned off. If you need to specify a number key from the numeric keypad, use the scan code shown for the keypad and the ASCII code shown for the corresponding typewriter key. For example, the keypad **7** has a scan code of 71 (the np Home scan code) and an ASCII code of 54 (the ASCII code for the number 7).

The chart is blank for key combinations that do not have scan codes or ASCII codes, like **Ctrl-1** or **Alt-PgUp**.

Top Keyboard Row

Key Cap	Scan	ASCII	Shift-Scan	Shift-ASCII	Ctrl-Scan	Ctrl-ASCII	Alt-Scan
Esc	1	27	1	27	1	27	1
~	41	96	1	126			41
1	2	49	2	33			120
2	3	50	3	64	3	0	121
3	4	51	4	35			122
4	5	52	5	36			123
5	6	53	6	37			124
6	7	54	7	94	7	30	125
7	8	55	8	38			126
8	9	56	9	42			127
9	10	57	10	40			128
0	11	48	11	41			129
-	12	45	12	95	12	31	130
=	13	61	13	43			131
Backspace	14	8	14	8	14	127	14

Second Keyboard Row

Key Cap	Scan	ASCII	Shift-Scan	Shift-ASCII	Ctrl-Scan	Ctrl-ASCII	Alt-Scan
Tab	15	9	15	0	148	0	165
Q	16	113	16	81	16	17	16
W	17	119	17	87	17	23	17
E	18	101	18	69	18	5	18
R	19	114	19	82	19	18	19
T	20	116	20	84	20	20	20
Y	21	121	21	89	21	25	21
U	22	117	22	85	22	21	22
I	23	105	23	73	23	9	23
O	24	111	24	79	24	15	24
P	25	112	25	80	25	16	25
[	26	91	26	123	26	27	26
]	27	93	27	125	27	29	27
\	43	92	43	124	43	28	43

Third Keyboard Row

Key Cap	Scan	ASCII	Shift-Scan	Shift-ASCII	Ctrl-Scan	Ctrl-ASCII	Alt-Scan
A	30	97	30	65	30	1	30
S	31	115	31	83	31	19	31
D	32	100	32	68	32	4	32
F	33	102	33	70	33	6	33
G	34	103	34	71	34	7	34
H	34	104	35	72	35	8	35
J	36	106	36	74	736	10	36
K	37	107	37	75	37	11	37
L	38	108	38	76	38	12	38
;	39	59	39	58			39
'	40	39	40	34			40
Enter	28	13	28	13	28	10	28

Bottom Keyboard Row

Key Cap	Scan	ASCII	Shift-Scan	Shift-ASCII	Ctrl-Scan	Ctrl-ASCII	Alt-Scan
Z	44	122	44	90	44	26	44
X	45	120	45	88	45	24	45
C	46	99	46	67	46	3	46
V	47	118	47	86	47	22	47
B	48	98	48	66	48	2	48
N	49	110	49	78	49	14	49
M	50	109	50	77	50	13	50
,	51	44	51	60			51
.	52	46	52	62			52
/	53	47	53	63			53
Space	57	32	57	32	57	32	57

Function Keys

Key Cap	Scan	ASCII	Shift-Scan	Shift-ASCII	Ctrl-Scan	Ctrl-ASCII	Alt-Scan
F1	59	0	84	0	94	0	104
F2	60	0	85	0	95	0	105
F3	61	0	86	0	96	0	106
F4	62	0	87	0	97	0	107
F5	63	0	88	0	98	0	108
F6	64	0	89	0	99	0	109
F7	65	0	90	0	100	0	110
F8	66	0	91	0	101	0	111
F9	67	0	92	0	102	0	112
F10	68	0	93	0	103	0	113
F11	133	0	135	0	137	0	139
F12	134	0	136	0	138	0	140

Numeric Key Pad

Key Cap	Scan	ASCII	Shift-Scan	Shift-ASCII	Ctrl-Scan	Ctrl-ASCII	Alt-Scan
/	224	47	224	47	149	0	164
*	55	42	55	42	150	0	55
-	74	45	74	45	142	0	74
7	71	0	71	55	119	0	
8	72	0	72	56	141	0	
9	73	0	73	57	132	0	
+	78	43	78	43	144	0	78
4	75	0	75	52	115	0	
5	76	0	76	53	143	0	
6	77	0	77	54	116	0	
1	79	0	79	49	117	0	
2	80	0	80	50	145	0	
3	81	0	81	51	118	0	
0	82	0	83	46	147	0	
Del	83	0	83	46	147	0	
Enter	224	13	224	13	224	10	166

Cursor Key Pad

Key Cap	Scan	ASCII	Shift-Scan	Shift-ASCII	Ctrl-Scan	Ctrl-ASCII	Alt-Scan
Home	71	224	71	224	119	224	151
Up	72	224	72	224	141	224	152
PgUp	73	224	73	224	132	224	153
Left	75	224	75	224	115	224	155
Right	77	224	77	224	116	224	157
End	79	224	79	224	117	224	159
Down	80	224	80	224	145	224	160
PgDn	81	224	81	224	118	224	161
Ins	82	224	82	224	146	224	162
Del	83	224	83	224	147	224	163

10.6.4 Keys & Key Names

Key names are used by ***4NT*** and ***TC*** to define keystroke aliases 187, in several .INI 91 directives, and in the KEYSTACK 296 command. The format of a key name is the same in all three cases:

```
[Prefix-]Keyname
```

The valid prefix and keyname combinations are shown in the table below. Names of keys must be spelled exactly as shown, except for case. Note that you cannot specify a punctuation key.

Prefix	Valid for keynames
none	A-Z, 0-9, F1-F12, Tab, Bksp, Enter, Up, Down, Left, Right, PgUp, PgDn, Home, End, Ins, Del, Esc
Alt-	A-Z, 0-9, F1-F12, Bksp
Ctrl-	A-Z, F1-F12, Tab, Bksp, Enter, Up, Down, Left, Right, PgUp, PgDn, Home, End, Ins, Del
Shift-	F1-F12, Tab

The prefix and key name must be separated by a hyphen (**-**). For example:

```
Alt-F10 ctrl-bksp
```

Some keys are intercepted by Windows and are not passed on to the command processor. For example, **Alt-Tab**, **Alt-Esc** and **Ctrl-Esc** typically pop up a task list, or are used in switching among multiple tasks. **Alt-space** brings down a menu to control window size and position, etc. Keys which are intercepted by the operating system (including menu accelerators, i.e. **Alt** plus another key) generally cannot be assigned to aliases 187 or with key mapping directives 102, because the command processor never receives these keystrokes. However, KEYSTACK 296 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.

10.7 ANSI X3.64 Command Reference

4NT and ***TC*** include support for ANSI Std X3.64, allowing you to manipulate the cursor, screen color, and other display attributes through sequences of special characters embedded in the text displayed on the screen. These sequences are called "ANSI commands". (For a general description of this feature, see ANSI Support 40.) ***4NT*** and ***TC*** cannot provide ANSI X3.64 support to external

applications.

The command processor supports most common ANSI X3.64 screen commands, but does not provide the complete set of options supported by some operating system's ANSI X3.64 drivers (for example, ***4NT*** and ***TC*** do not support ANSI X3.64 key substitutions; that functionality is already provided with key aliases 160). This section is a quick reference to the ANSI X3.64 commands supported by ***4NT*** and ***TC***.

ANSI X3.64 support within the command processor can be enabled or disabled with the ANSI 105 directive in the INI file 91, the corresponding option on the Colors tab 154 of the configuration dialogs 151, or the SETDOS 354 /A command. You can test whether ANSI X3.64 support is enabled with the _ANSI 411 internal variable.

An ANSI X3.64 command string consists of three parts:

<ESC>[	The ASCII character ESC, followed by a left bracket. These two characters must be present in all ANSI X3.64 strings.
parameters	Optional parameters for the command, usually numeric. If there are multiple parameters, they are separated by semicolons.
command	A single-letter command. (Case sensitive!)

For example, to position the cursor to row 7, column 12 the ANSI X3.64 command is:

<ESC>[7;12H

The **parameters** part of this command is "7;12" and the **command** part is "H".

To transmit ANSI X3.64 commands to the screen you can use the ECHO 253 command. The ESC character can be generated by inserting it into the string directly (if you are putting the string in a batch file and your editor will insert such a character), or by using the internal "escape" character 69 (defaults: caret, [**^**]) followed by a lower-case "e".

For example, the sequence shown above could be transmitted from a batch file with either of these commands (the first uses an ESC character directly, represented below by "ESC"; the second uses ^e):

```
echo <ESC>[7;12H
echo ^e[7;12H
```

You can also include ANSI X3.64 commands in your prompt 329, using $e to transmit the <ESC> character.

Commands

The internal ***4NT*** and ***TC*** ANSI X3.64 interpreter supports the subset of X3.64 commands below. Variable parameters are shown in lower-case italics, e.g., ***row*** and ***attr***, and must be replaced with the appropriate decimal numeric value when using the commands. The default value for ***row***, ***rows***, ***col***, and ***cols*** is 1.

<ESC>[*rows***A**	Cursor up by *rows*
<ESC>[*rows***B**	Cursor down by *rows*
<ESC>[*cols***C**	Cursor right by *cols*
<ESC>[*cols***D**	Cursor left by *cols*
<ESC>[*row***;***col***H**	Set cursor position (top left is row 1, column 1)

<ESC>[2J	Clear whole screen
<ESC>[K	Clear from cursor to end of line
<ESC>[*row***;***col***f**	Set cursor position, same as "H" command
<ESC>[*attr1***;***attr2***;...m**	Set display attributes; see table of attribute values below
<ESC>[s	Save cursor position (may not be nested)
<ESC>[u	Restore saved cursor position

Display Attributes

The attribute values used for the **m** command are:

0 Restore all attributes to default
1 High intensity (bright) foreground color
2 Normal intensity foreground color
30..37 Foreground color
40..47 Background color

Foreground Code	Background Code	Color
30	40	Black
31	41	Red
32	42	Green
33	43	Yellow
34	44	Blue
35	45	Magenta
36	46	Cyan
37	47	White

Attribute settings are cumulative, and are independent of order (except code **0**, reset to default), and they can be combined into a single command (using the **;** concatenation operator), or split into separate commands.

Examples

Set bright red foreground without changing background:

```
echo %=e[31;1m
```

Set the display to bright cyan on blue, and clear the screen:

```
echo %=e[44;36;1m%=e[2J
```

Set up a prompt which saves the cursor position, displays the date and time on the top line in bright white on magenta, and then restores the cursor position and sets the color to bright cyan on blue, and displays the standard prompt:

```
prompt $e[s$e[1;1f$e[45;37;1m$e[K$d $t$e[u$e[44;36;1m$p$g
```

10.8 Colors, Color Names & Codes

You can use color names in several of the directives in the .INI file 91 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 command processor 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 directives such as ColorDIR 109, 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 216 command also supports the CMD.EXE style color specification ***bf***, where ***b*** and ***f*** are CMD.EXE's codes for background and foreground colors, respectively (shown in the CMD.EXE columns of the table below). The numeric values of these codes are the same as the JP Software codes, but they are represented in hexadecimal.

ANSI X3.64 color codes are also shown in the table. Note that X3.64 support for the *bright* attribute is restricted to foreground. Note that the color codes are decimal, and the codes for *background* colors are larger than those of the corresponding *foreground* colors by 10.

SCREEN COLOR		**JPSoft name**	**JP color codes** (decimal)		**CMD.EXE** codes* (hexadecimal)		ANSI X3.64 codes (decimal)	
normal	bright		normal	bright	normal	bright	foreground	background
black	gray	**BLA**ck	**0**	**8**	0	8	30	40
blue	blue	**BLU**e	**1**	**9**	1	9	34	44
green	green	**GRE**en	**2**	**10**	2	A	32	42
cyan	cyan	**CYA**n	**3**	**11**	3	B	36	46
red	pink	**RED**	**4**	**12**	4	C	31	41

magenta	magenta	**MAG**enta	**5**	**13**	5	D	35	45
brown	yellow	**YEL**low	**6**	**14**	6	E	33	43
white	white	**WHI**te	**7**	**15**	7	F	37	47

Note: The numeric values of the CMD.EXE and native color codes are identical, the difference is in representation only.

Use one number to substitute for the **[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 482 function returns the value of color directives 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
```

(the line numbers are not part of the batch file and are merely to clarify any possible confusion caused by wrapping in your viewer).

Color Errors

A standard color specification allows sixteen foreground and sixteen background colors. However, most video adapters and monitors do not provide true renditions of certain colors. For example, most users see normal "yellow" as brown, and bright yellow as yellow; many also see normal red as red, and "bright red" as pink. Color errors are often worse when running in windowed mode, because Windows may not map the text-mode colors the way you expect. These problems are inherent in the monitor, video adapter, and driver software, and they cannot be corrected using the command processor color specifications.

11 Glossary

The glossary contains basic definitions for over 200 common terms listed in alphabetical order and is divided into sections by the first letter of each term. Concepts directly relevant to our command processors are typically discussed in more detail in other areas of this help file.

Select the glossary section you wish to review:

11.1 Glossary - 4

4EXIT - A program which is executed whenever ***4NT*** exits. See Automatic Startup and Termination Programs 8 for more details.

4START - A program which is executed whenever ***4NT*** starts. See Automatic Startup and Termination Programs 8 for more details.

11.2 Glossary - A

Access Date - In the Windows file systems it is the later of the dates on which a file was created or last accessed for reading. The **VFAT** 572 and **NTFS** 567 file systems save this file property.

Access Time - In the Windows file systems it is the time of day when the latter of the events of creating the file or last accessing the file for reading occurred. **NTFS** 567 saves this file property.

Advanced Power Management (**APM**) - A standardized system used by manufacturers of battery-powered computers to control system power management, including shutdown of unused components or of the entire system based on usage patterns. **APM** can also report the source of system power (AC or battery), the battery status, and the remaining battery life.

Age - A special term used by the command processor documentation to reference one of its methods to represent points in time (epochs). An **age** is the time elapsed since 1601-01-01 00:00:00 (local time) as a multiple of 100 ns. Internally the command processor uses 64-bit signed integers. Externally it is represented as any integer, by default a decimal integer. The dynamic range of **age** exceeds that of the arithmetic performed by the **@EVAL[ ]** function, so precision may easily be lost.

Alias - A command processor feature which allows you to create shorthand name for a command or series of commands. For details see Aliases 160.

Alias Argument - Same as Alias Parameter 555.

Alias Parameter - A numbered variable (*e.g.* %2) included in an alias definition, allowing a different value to be used in the alias each time it is executed.

Alternate File Name - Same as Short File Name 570.

American National Standards Institute (**ANSI**) - An organization which sets voluntary standards for a large variety of industrial products from boilers to photographic films, including computer-related systems, and is the U.S.A. representative in ISO 566. It is composed of various professional organizations, including EIA and IEEE, many of which also define standards.

AND - A logical combination of two true or false conditions. The result is **true** if and only if both conditions are true, otherwise it is **false.** See the table below.

condition 1	condition 2	result
false	false	false
false	true	false
true	false	false

true	true	true

ANSI - American National Standards Institute 555. The acronym **ANSI** in software development is often used as a short-hand reference to the standard ANSI X3.64 556

ANSI X3.64 - A standard specifying sequences of characters which control colors on the screen, manipulate the cursor and screen contents, and redefine keys. ***4NT*** and ***TC*** include support for a selected subset. This support may be enabled or disabled at any time. For more details see ANSI X3.64 Command Reference 550.

Append - Concatenation of one file or string onto the end of another.

APM - See Advanced Power Management 555.

Application - A program run from the command prompt or a batch file. Used broadly to mean any program other than the command processor itself; and more narrowly to mean a program with a specific purpose such as a spreadsheet or word processing program, as opposed to a utility.

Archive :

1) A file attribute indicating that the file has been modified since the last backup (most backup programs clear this attribute).
2) A single file (such as a ***.ZIP*** file) which contains a number of other files, optionally in compressed form.

Argument - Same as Parameter 568.

ASCII - Acronym for *American National Standard Code for Information Interchange*, defined in the standard **ANSI** 556 INCITS 4-1986 (R1997) Information Systems - Coded Character Sets - 7-Bit American National Standard Code for Information Interchange (7-Bit ASCII) (formerly ANSI X3.4-1986 (R1997)). This standard specifies a set of 128 characters (control characters and graphics characters such as letters, digits, and symbols) with their coded representation for use in information interchange among information processing systems, communication systems, and associated equipment. See ASCII file 556.

ASCII File - A file containing ASCII text, as opposed to a binary file which may contain numbers, or other information that cannot be sensibly interpreted as text. Often used in place of text file 571.

Attribute - A property, or indication of a property, of a file which may be set (present) or cleared (absent). Most attributes can be changed, but some cannot. The standard modifiable attributes are Read-Only 569, Hidden 565, System 571, and Archive 556. Unmodifiable attributes include Directory 561 and Volume Label 572.

Automatic Directory Change - A command processor feature which allows you to change directories by typing the directory name terminated in a backslash [\] at the command prompt, without first entering a command.

Automatic Programs - Programs automatically executed when the OS starts, or the command processor starts or terminates: **4START** 555, **4EXIT** 555, **TCSTART** 571, and **TCEXIT** 571. See Automatic Startup and Termination Programs 8 for more details.

11.3 Glossary - B

Base Name - The file name without a drive, path, or extension. For example, in the file name ***C:\DIR1\LETTER.DAT*** the base name is ***LETTER***.

Basic Input Output System (**BIOS**) - Software which provides basic low-level control of devices required to operate the system, such as the keyboard, floppy disk, and screen. On most systems the BIOS is stored in read-only memory inside the PC.

BAT File - Same as Batch File 557.

Batch File - A text file 571 containing a sequence of commands for the command processor to execute. Batch files are used to save command sequences so that they can be reexecuted at any time, transferred to another system, etc. The extension of a batch file may be ***.BAT***, ***.CMD***, or ***.BTM***, depending on the operating system and command processor you are using.

Batch File Argument - Same as Batch File Parameter 557.

Batch File Parameter - A numbered variable (*e.g.* %2) used within a batch file 556, allowing a different value to be used at that spot in the file each time it is executed.

Batch Program - A more descriptive name for Batch File 557.

Binary File - A file containing information which does not represent or cannot sensibly be interpreted as text. See also ASCII File 556, text file 571.

BIOS - Basic Input Output System 557..

Block Device - A physical device for input or output, which interchanges data with the computer in large blocks. XXIth Century PCs can perform such transfers of data concurrently with other activities. Examples of block devices include disk, tape, and CD-ROM drives. See also Character Device 558.

Boot - Ungrammatical contraction of Booting 557, shortened from Bootstrapping 557.

Boot Directory - The current directory at the time the system is started, usually the root directory 570 of the boot drive 557.

Boot Drive - The disk drive that the operating system is loaded from during the boot process, usually A: (the floppy disk) or C: (the hard disk), or on some systems a CD or DVD drive. In some diskless workstations it may be a network drive.

Booting - shortened form of Bootstrapping 557.

Bootstrapping - From the book "Baron Munchausen's Narrative of his Marvellous Travels and Campaigns in Russia", by Rudolf Erich Raspe (London, England, 1786), in particular, the story of the Baron's astounding ability to pull himself from the ocean by his own bootstraps. See also Cold Boot 558, Cold Reboot 558, Reboot 569, and Warm Reboot 573.

Break - A signal sent to a program to tell it to halt what it is doing. The **Ctrl-C** key or **Ctrl-Break** key is used to send this signal. Some external commands abort when they receive a break signal; others return to a previous screen or menu, or abort the current operation.

BTM File - A special type of batch file 557, unique to JP Software products, which is loaded into a buffer 557 in memory to speed up its execution.

Buffer - An area of memory set aside for temporary storage of information moved between concurrent programs or between programs and external media. For example, disk buffers are used to save information as it is transferred between your program and the disk, and the keyboard buffer

holds keystrokes until a program can use them. The command processor uses many buffers for its internal operations, e.g., to store aliases 160.

11.4 Glossary - C

Carriage Return (**CR**) - A control character (ASCII 541: 13) specifying that the next character is to be displayed at the leftmost position of the current line. In older mechanical character printing devices the *carriage* was moved to cause printing in different character positions of the same line. In a PC the **CR** code is generated by pressing the **Enter** key on the keyboard, and stored in most text file 571s at the end of each line. See also ASCII 541, End of Line 562.

CD-ROM File System (**CDFS**) - The file system which supports CD-ROM / CD-RW drives.

Central Processing Unit (**CPU**) - The part of the computer which historically performed all logic and most calculations. In PC-compatible systems, the CPU is on a single microprocessor chip. Many high-performance computers, including PCs, have more than one processing unit, often organized in such a way that none qualifies as the "central" processing unit.

Character Code - The code representing a specific character in the computer. See also: ASCII 556, Code Page 558, Unicode 572.

Character Device - A physical device for input or output which must communicate with your computer one character at a time. Examples include the console, communications ports, and printers. See also Block Device 557.

Character Mode - A display mode in which output is displayed one character at a time, and which cannot display graphics or pictures not included in the character font used. **Character Mode** displays are usually in a fixed font, typically with 80 columns in a line and 25 lines on the screen. Some systems allow you to increase or decrease the number of rows and columns to other fixed sizes. See also Graphic Mode 565.

CMD File - A batch file 557 designed for execution by **CMD.EXE**, **4NT**, or **TC.**

CMDLINE - An environment variable 562 used to extend the command line passed to another program beyond its normal length limits.

Code Page - A setting which tells Windows which character set to use, and how to retrieve and display date, time, and other information in the format appropriate to a particular country or region. See also Country Settings 560.

Cold Boot - The process of starting the computer by turning on its power. See also Boot 557, Cold Reboot 558, Reboot 569, Warm Reboot 573..

Cold Reboot - The process of restarting the computer in a way that physically resets most hardware devices, typically by pressing a reset button, or by emulating it utilizing a special feature of the BIOS 557. See also Boot 557, Cold Boot 558, Reboot 569, Warm Reboot 573.

Command Completion - A command processor feature which allows you to recall a previous command by typing the first few letters of the command, then an up-arrow or down-arrow.

Command Echoing - A command processor feature which displays commands as they are executed. Echoing can be turned on and off.

Command Grouping - A command processor feature which allows you enclose a group of several

commands within parentheses, and have them treated as a single command.

Command History - A command processor feature which retains the commands you have executed from the command prompt, so that they can be recalled, optionally modified, and reexecuted later.

Command History Window - A pop-up window used by the command processor to display the command history 559, allowing you to choose a previous command to modify and/or execute.

Command Interpreter - Same as Command Processor 559**.**

Command Line Expansion - The process the command processor goes through when it scans a command line and substitutes the appropriate actual values for aliases, alias parameters, batch file parameters, user defined functions, and variables. See also Parsing 568.

Command Processor - A program which interprets commands entered interactively or stored in a file (referred to as a batch file 557 in the environment discussed here), and executes other programs. Also called Command Interpreter 559.

Command Prompt - A prompt displayed on the screen by the command processor when it is ready for another command to be entered and executed.

Command Recall - See Command History 559.

Command Separator - One or more characters used to separate multiple commands on the same command line. See Multiple Commands 65.

Command Tail - The contents of the command line after removing the command name. See Separating the command from its command tail 70

Compound Command - See Multiple Commands 65.

Compression - A feature which compresses data as it is stored in a disk file, and expands it as it is read back, resulting in more efficient use of disk space. This feature may be provided by hardware, software, or a combination of both. The software may, but need not, be part of the operating system. NTFS 567 provides an attribute to indicate whether or not a file is compressed. Accessing compressed files requires slightly more processor time to perform the compression and expansion, but this is often compensated by the reduced data transfer time.

COMSPEC - An environment variable 562 which defines where to find the character-mode command processor to start a secondary shell. It is always set by ***4NT*** and ***TC***.

Condition - short for conditional expression 53.

Conditional Commands - A command processor feature allowing commands to be executed or skipped depending on the results of a previous command. See also: Exit Code 563.

Conditional Expression - See Conditional Expression 53.

Console - The PC keyboard and display.

Console Mode - See Character Mode 558.

Control Character - A character which does not have a printable or displayable representation, but, when encountered in a stream of characters, performs a specific action. In ASCII 541 the codes for

these characters are in the range [0,31] or it is 127; each character also has a two- or three-character symbol for representing it in plain text. The common PC keyboard for U.S. use has special keys for some, i.e., **ESC**, **CR**, **BS**, **HT**, **DEL**. Each control character in ASCII 541 can be generated by pressing the **Ctrl** key simultaneously with another key. For more information see ASCII, Key Codes & ANSI Commands 540.

Country Settings - The internal settings which tell the operating system how to interpret keyboard characters which vary from country to country, which character set to use, and how to retrieve and display date, time, and other information in the format appropriate to a particular country or region. See also Code Page 558.

CPU - Central Processing Unit 558.

CR - Carriage Return 558

CRC - Cyclic Redundancy Code 560.

Critical Error - An error, usually related to a physical or hardware problem with input, output, or network access, which prevents a program from continuing.

Current Directory - The directory in which all file operations of a selected drive will take place unless otherwise specified by an explicit path. The current directory is typically displayed as part of the command prompt. Also called the **Current Working Directory**.

Current Drive - The disk drive on which all file operations will take place unless otherwise specified. The current drive is typically displayed as part of the command prompt.

Current Working Directory - Alternate name for Current Directory 560.

Cursor - A movable marker on the screen to show where text will be entered when you type at the keyboard, or which object on the screen will be affected when a mouse button is clicked. In character mode only the text cursor is available; graphical systems typically show both a mouse cursor and, when text can be entered, a separate text cursor.

Cyclic Redundancy Code (**CRC**) - A sequence of characters, associated with a block of data, which provide a reasonably unique signature, used to detect the possibility of the data being corrupted in storage or transmission.

11.5 Glossary - D

Date Range - A command processor feature which allows you to select files based on the date and time they were last modified. For details, see Date Ranges 24.

Date Stamp - Information which may be stored in a file's directory entry to show the dates on which the file was created, last modified, and last accessed for reading. Only the modification date is available in the FAT file system 563. See also Time Stamp 525.

Default Directory - Same as Current Directory 560.

Default Drive - Same as Current Drive 560.

Deletion Tracking - An operating system or utility software feature designed to allow you to "undelete" or recover files which have recently been deleted. Delete tracking typically works by temporarily retaining the deleted files and / or information about the deleted files in a special area of

the disk.

Description - A special feature of JP Software command processors. A string of characters may be assigned to describe a file, using the **DESCRIBE** 230 command, typically stored in the file **DESCRIPT.ION** in the same directory as the file itself.

Destination - In some file processing commands (*e.g.* **COPY** 216 or **MOVE** 308), the name or directory the files should have after the command is completed. It is generally the last specification on the command line. See also Source 571.

Detached Process - A program which is detached from the normal means of user input and output, and cannot use the keyboard, mouse, or video display.

Device Driver - A program which allows the operating system to communicate with a device. It must be loaded into memory before the system can access the attached devices. Device drivers are also used to manage memory or for other similar internal functions.

Device - A physical device for input or output such as the console, a communications port, or a printer. Sometimes *device* is used to refer to character devices 558, and excludes block devices 557.

Directive - The name of a configurable command processor option and the value assigned to it. All directives may be stored in an Initialization File 565. Most directives may also be effectuated using the **OPTION** 317 command. See also: initialization file 565.

Directory :

1) A portion of a disk, identified by a name and a relationship to other directories in a directory tree 561 structure, with the tree starting at the root directory 570. A **directory** separates files on the disk into logical groups, but does not represent a physical division of the data on the disk.

2) The attribute 556 of the entry for the **directory** in its parent directory 568.

Directory History - A command processor feature which allows you to recall recently-used directory names in a popup window, and choose one to switch to.

Directory History Window - The name of the pop-up window displaying the Directory History 561.

Directory Junction - same as soft link 570.

Directory Stack - A command processor feature, implemented through the **PUSHD** 331 and **POPD** 326 commands, which allow you to save the current directory and return to it later. See also Stack 571.

Directory Tree - The branching structure of directories on a disk, starting at the root directory 570. The *root* of the tree is usually considered as the *top* of the structure, so the actual structure can be visualized as an upside-down tree with the root at the top and branches going down.

Domain -in mathematics the set of values for which a function is defined.

Drive Letter - A letter used by some operating systems to designate a specific local disk volume, or part or all of a network server volume. In most cases drive letters range from A - Z, but some network operating systems allow the use of certain other characters as drive letters to support more than 26 disk volumes.

11.6 Glossary - E

Echo - See Command Echoing 558.

End of Line (**EOL**) - An indication of the end of line in text file 571s. Many conventions exist to represent EOL in ASCII files. The most common ones use one or two control characters: (1) CR 558, (2) CR followed by LF 566, (3) LF followed by CR, (4) LF. Windows text files normally use (2). Unix/Linux and their variants use (4), and refer to the LF character as the new line (NL) character.

Environment - A set of variables and their values. Some variables are set before the start of the command processor. Others may be set at the command line using the **SET** 351 command, or by your batch files. If a variable is defined, it has a value, which is a character string, and is at least 1 character long. A variable can be removed from the environment using the **UNSET** 388 command, or by setting its value to an empty string. See also Master Environment 567 and Passed Environment 568.

Environment Variable - The name of a single entry in the environment 562.

Error Level - A numeric value returned from an external command or, in some cases, from an internal command to indicate its result (*e.g.*, success, failure, response to a question). Not all commands return an error level. Also known as Exit Code 563.

Escape Character -

1) The command processor escape character, used to suppress the normal meaning of, or to give special meaning to the immediately following character. The default **escape character** in ***4NT*** and ***TC*** is the caret (**^**). This may be modified using the EscapeChar 117 directive 561 or by using the /E option of the SETDOS 354 command.

2) A control character, symbol **ESC** (ASCII 541: 27).

Escape Sequence - A sequence of text characters which has a special meaning and is not treated as normal text. For example, the character sequence **<ESC>[K** (where **<ESC>** represents the ASCII "escape" character, decimal value 27) will cause an ANSI X3.64 driver to clear the screen from the cursor to the end of the current line, rather than simply displaying the string **ESC[K** on the screen. Similarly, in the command processor, the escape sequence **^f** on the command line is translated to a form feed, and is not treated as the literal characters **^f**.

Executable Extension - A command processor feature which allows you to specify the application to be executed when a file with a particular extension is named at the command prompt.

Executable File - A file, usually with the extension **.COM** or **.EXE**, which can be loaded into memory and run as a program.

Exclusive OR or **XOR** - A logical combination of two true or false conditions. If both conditions are false or both conditions are true the result is **false**; if either condition is true and the other is false the result is **true**. See below.

condition 1	*condition 2*	*result*
false	false	false
false	true	true
true	false	true
true	true	false

Exit Code - The result code returned by an external command or an internal command. Internal commands return an exit code of 0 if successful, or non-zero if unsuccessful. External commands may, but need not return an exit code. See also Errorlevel 562.

Expansion :

1) Command Line Expansion 559
2) The inverse of compression 559.

Extended ASCII Character: A legally invalid phrase, used for a character whose code is in the range 128 to 255, used on the PC as part of an extended set of 256 characters. The extension character set may include international language symbols, and box and line drawing characters. What is displayed or what is printed when such a character is used depends on the country setting 560, code page 558, and font.

Extended Directory Search - A command processor feature which maintains a directory search database or list, typically including all directories in your system, and allows you to change quickly to any directory in the list based on partial match of the directory you specify in the command.

Extended Key Code - The code for a key on the PC keyboard which has no representation in the standard ASCII character set, such as a function key, cursor key, or **Alt** plus another key. The extended key code for a key is often the same as the scan code for that key. See Key Codes and Scan Codes Table 545.

Extended Parent Directory Names - A command processor feature which allows you to use additional periods in a directory name to represent directories which are successively higher in the directory tree.

Extended Wildcard - A command processor feature which extends the wildcard syntax and allows you to use multiple wildcard characters, and character ranges (*e.g.* **`[a-f]`** for the letters A through F). See also Wildcard 573.

Extension - The portion of a file name following the last period. For example, in the file name ***`C:\DIR1\LETTER.JOHN.DAT`*** the extension is ***`.DAT`***.

External Command - A program directly executable by the operating system, as distinguished from an internal command 566, which is executed by the command processor.

EXTPROC - A command processor feature which allows you to designate a specific external program to run a particular batch file instead of the command processor.

11.7 Glossary - F

FAT - File Allocation Table 564.

FAT-Compatible File Name - See SFN 570.

FAT File System - The file system used by PC-DOS, MS-DOS, and its emulators based on FAT 564 .to store files on diskettes and hard disks; also supported by Windows. Supports a single 2-s resolution timestamp of "last written" (modified). It is subclassified according to the size of word required to represent the largest possible FAT 564 size. Three versions have been implemented: 12-bit 563, 16-bit 564,and 32-bit 564.

FAT12 File System - Each entry in the FAT 564 is 12 bits long, limiting table size to 1,024 allocation

blocks.

FAT16 File System - Each entry in the FAT 564 is 16 bits long, limiting table size to 65,536 allocation blocks.

FAT32 File System - Each entry in the FAT 564 is 32 bits long, limiting table size to 4,294,967,296 allocation blocks, See also VFAT File System 572.

FF - Form Feed 564.

File Allocation Table (**FAT**) - A table used by the file system to keep track of disk space usage, allocation, ownership, and file content.

File Attribute - See Attribute 556.

File Description - See Description 561.

File Exclusion Range - A command processor feature which allows you to exclude files from processing by internal commands. See File Exclusion Ranges 27.

Filename Completion - A command processor feature which allows you to type part of a filename on the command line, and have the command processor fill in the rest for you.

Font - a set of character shapes in a single style and size.

Form Feed (**FF**)- A control character (ASCII: 12), which typically causes a printer to skip to a new page. The **FF** character is not normally entered from the keyboard, but in many cases it can be generated, if necessary, by holding the **Alt** key, pressing the numeric pad keys **012**, and releasing the **Alt** key, or by the **Ctrl-L** key combination.

Free Memory - Usually, the amount of total memory which is unoccupied and available for applications.

11.8 Glossary - G

Global Aliases - A command processor option which allows you to store aliases in a global area accessible to all copies of the command processor, so that any change made by one copy, even between a **SETLOCAL** 358 - **ENDLOCAL** 256 command pair, is immediately effective in all other copies. See also Local Aliases 567..

Global Directory History - A command processor option which allows you to store the directory history in a global area accessible to all copies of the command processor, so that any change made by one copy, even between a **SETLOCAL** 358 - **ENDLOCAL** 256 command pair, is immediately effective in all other copies. See also Local Directory History 567.

Global History - A command processor option which allows you to store the command history in a global area accessible to all copies of the command processor, so that any change made by one copy, even between a **SETLOCAL** 358 - **ENDLOCAL** 256 command pair, is immediately effective in all other copies. See also Local History 567.

Global User Functions - A command processor option which allows you to store user-defined functions in a global area accessible to all copies of the command processor, so that any change made by one copy, even between a **SETLOCAL** 358 - **ENDLOCAL** 256 command pair, is immediately effective in all other copies. See also Local User Functions 567.

Graphics Mode - A display mode in which output is displayed in any one of a range of fonts, typically in resizable windows with a variable number of text rows and columns, and which supports the display of graphics and pictures along with text. See also Character Mode 558.

11.9 Glossary - H

Hard link - NTFS gives you the ability to create hard links, which are multiple directory entries referencing a single file body. Creating a hard link causes the system to create an additional directory entry that points to the same file (unlike a shortcut, which is actually an additional file). Files can have multiple hard links, and therefore a single file can appear in multiple directories, with multiple names in each.

A file with more than one hard link can be accessed by any of its directory entries. Each file on an NTFS volume has at least one hard link (connecting a directory entry to the file body). A file is deleted from the file system only after all its hard links (i.e., directory entries) have been deleted.

When a file with more than one directory entry is modified, the file properties (e.g., size, date stamps, etc.) of only the entry actually used for updating the file are updated instantly.

Hard links must be created on the same NTFS volume as the file.

Hidden - A file attribute indicating that the file should not be displayed with a normal DIR 233 command, and should not be made available to programs unless they specifically request access to hidden files. Often combined with the System 571 attribute.

History - See Command History 559.

History Window - See Command History Window 559 and Directory History 561.

11.10 Glossary - I

IFS - Installable File System 565.

Installable File System (**IFS**) - A file system for which device drivers can be loaded when required to support devices such as CD-ROM or network drives, or non-default disk formats. Installable file systems may be loaded at system startup, or loaded and unloaded dynamically while the system is running. They typically appear to the command processor simply as yet another drive .

Include List - A method of specifying several files or groups of files in the same directory, for use with all internal commands which take file names as parameters.

Inheritance - A command processor feature which allows one instance of the command processor to "inherit" the .INI file data, aliases, user defined functions, environment variables, command history, and directory history from a previous instance. More generally, a system which allows one program to pass information or settings on to another, often to a second instance of the same program.

Initialization Directive - See Directive 561.

Initialization File or **.INI File** - A file which enumerates the initial settings of various command processor options, and which is automatically read by the command processor when it starts. The default initialization files are **4NT.INI** for ***4NT*** and **TCMD32.INI** for ***TC***. For additional details, see Startup Command 4 and Initialization Files 91.

Insert Mode - When editing text, a mode in which newly typed characters are inserted into the line at the cursor position, rather than overwriting existing characters on the line. See also Overstrike Mode 568.

Internal Command - A command which is part of the command processor. See also external command 563..

Internal Variables - Special variables created by the command processor to provide information about your system. Internal variables are evaluated each time they are used, and are not actually stored in the environment. Internal variables are accessed using the same syntax as environment variables.

International Standardization Organization (ISO) - An international body promulgating standards.

ISO - International Standardisation Organization 566.

11.11 Glossary - J

Junction - same as soft link 570

11.12 Glossary - K

Key Code - The code passed to a program when a key is pressed on the keyboard. Depending on the key that is pressed, and the software handling the keyboard, the code can be an ASCII value, a scan code, or an extended key code.

Key Mapping - A command processor feature which allows you to assign new keystrokes for command line functions such as manipulating the command history or completing file names.

Keyboard Buffer - An operating system buffer which holds keystrokes you have typed that have not yet been used by the currently executing program.

Keystroke Alias - An alias assigned to a key so that it can be invoked or recalled with a single keystroke.

11.13 Glossary - L

Label :

1) A location marker in a batch file 557, with the format **:name**, allowing **CALL** 209, **GOTO** 282 and **GOSUB** 280 commands to "jump" to that point in the file.
2) Volume Label 572.

Line Feed (**LF**) - A control character (**ASCII** 541: 10) indicating that the next character should be displayed in the current horizontal position but one line down. It is often used in text files as part of, or as the End of Line 562. It is not normally entered from the keyboard, but in many cases it can be generated, if necessary, by pressing **Ctrl-J**.

LF - Line Feed 566.

LFN - Long File Name 567.

LFN File System - A file system which supports long file names 567. An LFN file system may store both a long and short name for a file. The short name is sometimes called the alternate name. See

also Long File Name 567, Short File Name 570, VFAT File System 572, and NTFS 567.

Local Aliases - A command processor option which allows you to store aliases in a local area only accessible to the current copy of the command processor, so that a change made in the current copy of the command processor does not affect other copies, and vice versa. See also Global Aliases 564.

Local Directory History - A command processor option which allows you to store the directory history in a local area only accessible to the current copy of the command processor, so that a change made in the current copy of the command processor does not affect other copies, and vice versa. See also Global Directory History 564.

Local History - A command processor option which allows you to store the command history in a local area only accessible to the current copy of the command processor, so that a change made in the current copy of the command processor does not affect other copies, and vice versa. See also Global History 564.

Local User Functions - A command processor option which allows you to store user-defined functions in a local area only accessible to the current copy of the command processor, so that a change made in the current copy of the command processor does not affect other copies, and vice versa. See also Global User Functions 564.

Logging - A command processor feature, implemented via the **LOG** 303 command, which allows you to save a record of the commands you execute.

Long File Name (**LFN**) - A file name which does not conform to FAT file system restrictions, either because it is longer than the 8 character name plus 3 character extension, or because it contains spaces, multiple periods, or other characters not allowed in a FAT file name. See also Short File Name 570.

11.14 Glossary - M

Master Environment - The master copy of the environment maintained by Windows.

Modulo - The remainder after an integer division. For example 11 modulo 3 is 2.

Multiple Commands - A command processor feature which allows multiple commands to be placed on a line, with each command separated by a command separator 559..

11.15 Glossary - N

Name - The part of the file name from its beginning up to, but not including, the last period in the file name.

Network - A system which allows several computers to be connected together to share files, printers, modems, or other resources, and to pass electronic mail or other information between the systems on the network.

Network File System - Software which runs over a network to allow access to files on the server. A network file system may support the same options as the file system used on local drives, or it may be more or less restrictive than the local file system about file names, disk volume capacity, and other similar features.

New Technology File System (**NTFS**): A file system distributed with Windows which allows longer

file names, supports larger drives, and provides better performance than the FAT file system 563.

11.16 Glossary - O

Operating System - A collection of software which provides access to various devices, including file systems and networks, services to other software, and ensures that programs don't interfere with each other while they are running.

Option :

1) A parameter for an internal command or application which specifies a particular behavior or setting. For example, the command "DIR /P" might be referred to as "having the /P option set". Alternate name: **switch**.
2) A command which allows modifying command processor operating characteristics.

Option file - Alternate name for Initialization File 565.

OR - A logical combination of two true or false conditions. If both conditions are false the result is false; if either condition is true the result is true. See the table below.

condition 1	*condition 2*	*result*
false	false	false
false	true	true
true	false	true
true	true	true

Overstrike Mode - When editing text, a mode in which newly typed characters overwrite existing characters on the line, rather than being inserted into the line at the cursor position. See also Insert Mode 566.

11.17 Glossary - P

Parameter - Additional information placed after a command or function name. For example, in the command **DIR** ***XYZ***, ***XYZ*** is a parameter. Also used to refer to an alias parameter or batch file parameter.

Parent Directory - The directory in which a particular subdirectory is cataloged, often seen as the directory above a subdirectory.

Parsing - The process the command processor performs to analyze the command line, perform alias, function and variable expansion, and find the appropriate internal command or external command to execute. More generally, the process of breaking down a string or message into its individual components in order to process them properly. See

Passed Environment - A copy of the environment created before running an application, so that any changes made by the application will not affect the environment of the application's parent.

Path :

1) A specification of all the directories required to locate a file. For example, the path for ***C:\WPFILES\MYDIR\MEMO.TXT*** is ***C:\WPFILES\MYDIR***.
2) The environment variable **PATH** 399, which contains a series of path specifications used when searching for external commands and batch files.

3) The internal command **PATH** 319, which redefines the value of the environment variable **PATH** 399.

Picture Element (**Pixel**) - A single display element on the screen. In graphic mode, the color of each picture element can be individually controlled.

Pipe - A method for sending the standard output of one command to the standard input of another command, syntactically represented by a vertical bar "**|**" separating the commands. See also Redirection 36.

Pixel - A contraction of Picture Element 569.

Previous Working Directory - The working directory used most recently, just prior to selecting the current working directory 560. For example, if ***C:\DATA*** is the current working directory and you switch to ***D:\UTIL***, ***C:\DATA*** becomes the previous working directory.

Primary Shell - The copy of the character-mode command processor which is loaded by the operating system when the system starts or a session opens.

11.18 Glossary - Q

Quote mark - The character **"** used by English and other natural languages to mark the beginning and end of a block of text copied from another source, and thus not subject to modification. Many programming languages, including ***4NT*** and ***TC***, use it in a similar manner to delimit text to be used literally (i.e., exactly as is).

11.19 Glossary - R

RAM - Random Access Memory 569.

RAM Disk - Another name for Virtual Disk 572.

Random Access Memory (**RAM**) - The physical memory used to store data while a computer is operating. The information in most types of RAM is lost when power is turned off.

Range (file selection) - See Date Range 24, Description Range 28, File Exclusion Range 27, Size Range 24, Time Range 26.

Range - the set of possible values of a function.

Read Only - A file attribute indicating that the file can be read, but not written or deleted by the operating system or the command processor unless special commands are used.

Read Only Memory (ROM) - A physical memory device used to store information which cannot be readily modified.

Reboot - The process of restarting the computer with software, with the keyboard (*e.g.* by pressing Ctrl-Alt-Del), by pressing a reset button, or by turning the power off and back on. See also Boot 557, Cold Reboot 558, and Warm Reboot 573.

Redirection - A method for sending the output from a command to a file, or of providing the input for a command from a file. See also Pipe 569.

Registry - A hierarchically organized data file (or set of files) maintained by Windows to hold system

parameters, hardware and software settings, and other similar information used by the operating system or by other software packages.

Reparse Point - same as soft link 570

REXX - A file and text processing language developed by IBM, and available on many PC and other platforms.

ROM - Read Only Memory 569.

Root Directory - The first directory on a disk, from which all other directories are "descended". The root directory is referenced with a single backslash [****].

11.20 Glossary - S

Scan Code - The physical code for a key on the keyboard. For the original U.S. English keyboard layout the scan code represents the physical position of the key, starting with 1 for the key in the upper left corner (Esc), and increasing from left to right and top to bottom. This order will vary for more recent keyboards or those designed for other countries or languages. See Key Codes and Scan Codes Table 545.

Search Path - A semicolon-delimited list of directories in which to search for a file.

Secondary Shell - A copy of the command processor which is started by another program, rather than by the operating system.

Section - A part of an Initialization File 565 starting with a section name enclosed in brackets, e.g., **`[primary]`**.

Session - A general term for the individual windows or tasks started by a multitasking system.

Shell - The name used in Unix/Linux/etc. systems for command processor 559, from a pictorial representation of the OS being encased by a shell.. Also used to refer to any program which gives access to operating system functions and features through a menu- or mouse-driven system, or which replaces the primary user interface of the operating system.

Short File Name (**SFN**) - A file name which follows the rules of the FAT file system 563: a name 567 of 0 ..8 characters and an extension 563 of 0 .. 3 characters, each consisting of only alphabetic and numeric characters plus the punctuation marks **! # $ % & ' ( ) -** @ **^** _ **`** **{ }** and **~**. See also **LFN** 567.

VFAT file systems consider a file name that contains a lower case character to be a LFN 566, even if it obeys all other rules to a SFN, and assign to it a SFN - the all upper case version of the LFN. NTFS does not assign an SFN to such files.

When you copy an LFN file to another LFN directory by its LFN (using the same name in the target directory), the SFN of the target file may be different from the SFN of the source file.

Size Range - A command processor feature which allows you to select files based on their size. For details see Size Ranges 24.

Soft Link - also known as a directory junction or reparse point - is a special file. When created, its content is the name of a directory on any volume. Any reference to the contents of the soft link act on the contents of the directory it references, including deleting of files from the soft link, which deletes the files from the referenced directory.

Soft links can be chained, i.e., a soft link **A** can refer to another soft link **B**. Changing the referenced directory of **B** effectively changes the contents of **A**.

If the directory referenced by a soft link is deleted, references to the soft link's contents fail. If later another directory is created with the same name, the soft link will function again. Similarly, if the reference is to a removable volume, it will reference whatever volume is present at the time of access.

Source - In file processing commands (e.g. COPY or MOVE), the original files before any copying or modification has taken place, *i.e.*, those specified earlier on the command line. See also Destination 561.

Stack - An area of memory used by any program to store temporary data while the program is running; more generally, any such storage area where the last item stored is normally the first one removed.

Standard Error (stderr)- The file or character device where a program displays error messages. It defaults to the screen. It may be redirected 36 to a file, or piped 39 to another program.

Standard Input (stdin) - The file or character device whence a program obtains its normal input. It defaults to the keyboard. It may be redirected 36 to a file, or piped 39 to another program.

Standard Output (stdout) - The file or character device where a program displays its normal output. It defaults to the screen. It may be redirected 36 from a file, or piped 39 from another program.

Stderr - short for Standard Error 571.

Stdin - short for Standard Input 571.

Stdout - short for Standard Output 571.

Subdirectory - Any directory other than the root directory 570.

Subtree - See Directory Tree 561.

Switch - Alternate name for Option 568; also the name of the program control command SWITCH 367.

System - A file attribute indicating that the file belongs to the operating system or command processor, and should not be accessed by other programs. Almost every file with this attribute also has the Hidden 565 attribute.

11.21 Glossary - T

Target: See Destination 561.

TCEXIT: A program which is executed whenever ***TC*** exits. See Automatic Startup and Termination Programs 8 for more details.

TCSTART: A program which is executed whenever ***TC*** starts. See Automatic Startup and Termination Programs 8 for more details.

Text file: A file containing only characters that are either displayable, or which affect the display format (format effectors). The characters of the file are represented by their character codes. ***4NT*** and ***TC*** provide support for two systems of character codes, ASCII 541 and Unicode 506.

Time Range: A command processor feature which allows you to select files based on the time they were last modified, created, or accessed. See Time ranges 26 for more details.

Time Stamp: Information stored in a file's directory entry to show the times at which the file was created, last modified, and last accessed. Creation time is not available in the FAT file system; last access time is only available in the NTFS file system. See also Date Stamp 560.

Tree: See Directory Tree 561.

11.22 Glossary - U

UNC - Universal Naming Convention

Universal Naming Convention - A common method for accessing files on a network drive without using a mapped drive letter. Names specified this way are called UNC names, and typically appear as \\server\volume\path\filename, where server is the name of the network server where the files reside, volume is the name of a disk volume on that server, and the path\filename portion is a directory name and file name.

UNICODE - A character set standard, designed and maintained by the non-profit Unicode Consortium (http://unicode.org). Windows 2000 introduced an implementation of 16-bit *UNICODE* (65536 potential distinct values) intended to facilitate use of non-English languages. The first two bytes of Unicode text files in NTFS systems contain a signature representing the file encoding.

UDF - User Defined Function.

User Defined Function - A function defined by the user utilizing the FUNCTION 275 command to manipulate strings, dates, and filenames; perform arithmetic; read and write files; and perform other similar functions. UDFs are invoked the same way as Variable functions 424.

11.23 Glossary - V

Variable - See Alias Parameter 555, Batch File Parameter, Function Parameter, and Environment Variable.

Variable Expansion - The process of scanning a command line and replacing each environment variable name, alias, function, and batch file parameter and function invocation with its value.

Variable Functions - Functions provided by the command processor to manipulate strings, dates, and filenames; perform arithmetic; read and write files; and perform other similar functions. Variable functions are similar to environment variables or internal variables, but have parameters and can perform actions rather than just returning static information.

VFAT File System - An extension of the FAT file system 563 which supports long filenames 567.

Virtual Disk - A portion of memory, dedicated for this use through special purpose software, accessible by programs as if it were a physical disk drive, and storing data in files in a directory tree. Also called a **RAM Disk**.

Volume - See **Disk Drive**.

Volume Label - A character string stored on a disk drive or volume used to identify it. Windows uses a special, hidden file name in the disk root directory for this purpose.

11.24 Glossary - W

Warm Reboot - The process of restarting the computer with software, or with the keyboard (e.g. by pressing Ctrl-Alt-Del), typically without physically resetting any hardware devices. See also **Cold Reboot**.

White Space Character - A character used to separate parameters on the command line. The white space characters recognized by the command processor are the space, tab, and comma.

Wildcard - A character (***** or **?**) used in a filename to specify the possibility that any single character (**?**) or sequence of characters (*****) can occur at that point in the actual name. See also **Extended Wildcard.**

11.25 Glossary - X

X-3.64 - See ANSI X-3.64 556

XOR - Exclusive OR 573.

12 What's New

```
NEW VERSION OVERVIEW -  Take Command and 4NT 8.0
-----------------------------------------------

Since the last major release we've added over 200 new features,
commands, switches, variables, and other enhancements to 4NT and
Take Command!

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


FEATURE LIST:
-------------

    The window menus are now in the language dll's, so they will be
      localized for English, German, and French.

    You can now specify regular expressions for file name tests.
      The syntax is:

        ::regex

      For example:

        dir ::ca[td]

    Changed the startup inheritance so that 4NT / TC will only inherit
      .INI values from a prior 4NT / TC shell if they're a child pipe
      process or were started from a parent 4NT / TC process.  (This
      also means the %_shell internal variable has changed; see below.)

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

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

  Then you can reference the files in "c:\program files\jpsoft" by
  entering "pf:\jpsoft".  Directory aliases work in places that
  accept filenames and directory names, including filename
  completion.

FTP / IFTP - added internal support for additional FTP LIST formats.
  The parser should now be able to recognize these formats:

    EPLF
    WFTPD
    VMS (single-line filenames only)
    NetPresenz (Macintosh)
    NetWare
    All known UNIX and Linux formats
    Windows FTP server

FTP - now supports FTP / FTPS accounts where you do not have
  access to the root (i.e., logon puts you into a subdirectory).
  (IFTP already supported this.)  Unfortunately, this also means
  that FTP access (not IFTP) is slightly slower.

FTP / IFTP - added support for symbolic hostnames (i.e., in
  "\windows\system32\drivers\etc\hosts"), and using symbolic
  names and port numbers with no username / password.

If you press Ctrl-C in Take Command, it now checks to see if it is
  at the prompt (where a "Copy" would be valid); if it is not, a
  Ctrl-C will act like a Ctrl-Break.

Added an additional parameter type for alias argument expansion:

  %-n$ - substitute all of the arguments from 1 to n-1.

Added an additional parameter type for batch variable expansion:

  %-n$ - substitute all of the arguments from 1 to n-1.

Added an additional parameter type for user-defined function
  argument expansion:

  %-n$ - substitute all of the arguments from 1 to n-1.

Added support for stream completion (i.e., "test:t<tab>").  Note
  that you cannot attempt to complete both a filename and a stream
  name at the same time (i.e., "t*:t<tab>" will NOT work!).

Added regular expression searches of file descriptions.  The syntax
  is:

  /R"text"

  The regular expression searches also support exclusions
  (/!R"text").

Added ! (exclusion) for /I (description) searches.  The syntax is:

  /!I"text"
```

Added ! (exclusion) for /[D..] (date) ranges. The syntax is:

/![D...]

For example, to exclude all of today's files:

/![D]

Added ! (exclusion) for /[S..] (size) ranges. The syntax is:

/![Sn]

For example, to exclude all files that are >= 100 bytes:

/![S100]

Added ! (exclusion) for /[T..] (time) ranges. The syntax is:

/![T...]

File exclusion ranges now allow directories to be excluded. For example:

dir /s /[!foo\] * - exclude any subdirectories named "foo" anywhere in the directory tree.

When at the command prompt (i.e., not executing a batch file) the parser will look for the following aliases:

PRE_INPUT - executed immediately before accepting input for a new command line.

PRE_EXEC - executed immediately after a command line is entered (before any expansion or redirection).

POST_EXEC - executed immediately after returning from a command and before displaying the prompt.

None of these aliases will be passed any arguments.

Plugins can now define functions that will receive every keystroke entered at the command prompt. A keystroke plugin can call other functions or even change the keystroke before passing it back to 4NT / TC.

Startup Options

There is a new startup switch for 4NT and Take Command:

/H - start 4NT or TC as a hidden window. Note that in 4NT you will still see the console window momentarily appear and then disappear (Windows limitation). With TC, you will not see any window appear. The 4NT / TC window will not appear on the start bar, or in the Alt-Tab list of applications -- if you need to kill it you'll have to use the Windows Task Manager.

Command Line Editing:

Added Shift-Ins as an alternative to ^V (paste from clipboard).

Filename completion now checks whether the filename is inside a variable function, so you no longer need to add a space following the left bracket (i.e., "%@name[a<tab>]" instead of "%@name[a<tab>]".)

New .INI directives:

AutoCancel=NO|yes - if set to YES, a ^C will cancel a batch file without any prompting.

BGColorRGB=n,n,n (TCMD only) - set the default background color as an RGB value.

FGColorRGB=n,n,n (TCMD only) - set the default foreground color as an RGB value.

InactiveTransparency=nnn (TCMD only) - set the transparency level of the TCMD window when it is out of focus (range is 40-255).

ListInverseColors=fg on bg (4NT only) - set the color to use in LIST when showing search matches.

LogAll (TCMD only) - log all output (see LOG below).

Modified .INI directives:

ColorDir - now supports and/or/xor tests for the attributes (see DIR).

Editor - now supported in 4NT. Editor also now supports variables and aliases in the Editor argument. (Note that this means if you have been specifying a LFN with embedded whitespace, you will now need to quote the program name.)

WindowState - added new option "Hidden" to hide the command processor window at startup. (See "Startup Options" above.)

New Internal Variables:

%_CDROMS - returns a space-delimited list of the CD-ROM drives.

%_DVDS - returns a space-delimited list of the DVD drives.

%_EDITMODE - returns 0 if in overstrike mode, or 1 if in insert mode.

%_EXECSTR - returns the return code of the last @EXECSTR function.

%_EXPANSION - returns the current expansion mode (SETDOS /X).

%_HDRIVES - returns a space-delimited list of the hard drives.

%_READY - like %_DRIVES, but returns a space-delimited list of the drives that are ready.

%_REGISTERED - returns the registered name or an empty string if 4NT / TC isn't registered.

%_SHELLS - returns the same value that %_SHELL did in previous versions (see below).

%_VIRTUALPC - returns 1 if 4NT / TC is running inside VirtualPC.

%_VMWARE - returns 1 if 4NT / TC is running inside VMWare.

%_WOW64 - returns 1 if 4NT / TC is running in the WOW64 environment.

Modified Internal Variables:

%_shell - will always now return 0 unless 4NT / TC was started by another 4NT / TC process (either directly or via a pipe.) This now matches the %_shell with the documentation; if you have need of the old %_shell behavior you can use %_shells.

New Variable Functions:

%@CAPI - like @WINAPI, but for _cdecl functions. The syntax is:

@CAPI[module,function[,integer | NULL | BUFFER | |PINT=n | PLONG=n | PDWORD=n "string"]
module - name of the DLL containing the function
function - function name (case sensitive)
integer - an integer value to pass to the function
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
PINT=n - pointer to an integer
PLONG=n - pointer to a long
PDWORD=n - pointer to a DWORD

%@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

    %@OWNER - returns the owner of the specified file (if any).

    %@SCRIPT - execute a line in the specified Active Scripting engine.
      The syntax is:

        %@script[engine,command]

      For example, to execute the command "MsgBox" in VBSCRIPT:

        %@script[vbscript,MsgBox "Called MsgBox from 4NT"]

      (You can find the names of the installed engines by running
      the SCRIPT command (see below) with no arguments.)

    @UNQUOTES - removes leading & trailing double quotes.

    @WINPOS - returns normal position of the window with the
      specified title in the format "top,left,bottom,right".

    @WMI - query Windows Management Interface.  This function is *very*
      powerful, letting you query nearly any Windows value.  The syntax
      is:

     %@WMI[namespace,"wql search string"[,enum]]


      You can omit the "enum" parameter if you're querying a single
      property and instance.  For example:

        %@wmi[root\cimv2,"SELECT name FROM Win32_Processor"]

      The optional "enum" parameter specifies the property instance to
        return (for classes that return multiple properties).

     echo %@wmi[root\cimv2,"SELECT name, state FROM Win32_service",4]

      For more details on what is available, see the WMI and WQL
      documentation on MSDN (msdn.microsoft.com), and download the
      "WMI Code Creator" from Microsoft and browse the available
      namespaces, classes, and properties.  Also see the new WMIQUERY
      internal command (below).

    @XMLPATH - return the text of the current element.  The syntax is:

      @XMLPATH[filename,path]

        filename - name of XML file
      path - one or more element accessors separated by a /.


Modified Variable Functions:
----------------------------

    %@EVAL - if a "=X" is appended to the expression, the output will
      be in hex with a leading "0x".
```

%@FIELD - added ranges (see @WORD for details).

%@TRUENAME - added support for junctions.

%@WINAPI - added new argument types to pass pointers to integers, longs, and dwords:

PINT=n
PLONG=n
PDWORD=n

%@WORD - added ranges. The syntax is:

%@WORD[start[-end|+range],string]

You can 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".

New Internal Commands:

OSD - (On Screen Display) Writes "floating" text to the display (like TV or monitor prompts). OSD has several options:

/Font=n	Font height (default 18)
/N	Don't wait for timeout
/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 (0-255) (default 0,255,0)
/TIME=n	Time to display the text in seconds (default 10)
/TOP	Position the text to 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

You can combine the window positioning options.

PRIORITY - display or set process priority, or suspend / resume process. The syntax is:

PRIORITY [/Q /R /S PID | "title" ABOVE | BELOW | NORMAL | HIGH | IDLE | REALTIME]

/Q - quiet (don't display resume/suspend messages)
/R - resume process
/S - suspend process
PID - process ID
title - main window title
ABOVE - above normal priority

```
        BELOW - below normal priority
        NORMAL - normal priority
        HIGH - high priority
        IDLE - idle priority (runs only when system is idle)
        REALTIME - realtime priority

      You can specify the process either by the PID or by the window
      title.  If you do not enter any arguments, PRIORITY displays the
      processes, their current priority, and the module names.  If you
      don't enter a window title or PID, PRIORITY will set the priority
      of the current process.

    SCRIPT - execute scripts using Active Scripting engines (i.e.,
      VBScript, JavaScript, PerlScript, etc.)  The syntax is:

        SCRIPT [/E engine] [filename ...]

          /E - specify the scripting engine to use (if SCRIPT can't
               determine it from the extension).

      If you don't specify any arguments, SCRIPT will display the names
      of the installed active scripting engines.

    SNMP - send SNMP traps.  The syntax is:

      SNMP remotehost trapOID "value" [username password]

        remotehost
        trapOID - the full OID of the trap
      value - descriptive text
      username - username for SNMPv3 trap
      password - password for SNMPv3 trap

      SNMP normall sends an SNMPv2 Trap; if you specify a username 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

    SYNC - synchronize two directories.  The syntax is:

      SYNC [/D /E /G /H /I"description] /K /N /P /Q /S[n] /V /W /X /Z]
source target

        /W - delete files in the target directory that are not present
       in the source directory.

        (The other SYNC options are the same as for COPY.)

    WMIQUERY - Query the Windows WMI interface.  The syntax is:
```

```
      WMIQUERY [/A /B /C /H] namespace "query string" [index]

        /A - display all class instances starting at "index"
        /B - separate class instances with a blank line
      /C - display matching classes for the specified namespace
             ("query string" is filter)
        /H - display a header for class instances
      namespace - a . defaults to root\cimv2
      "query string" - WMI
      index - instance of the class to display

        wmiquery . "SELECT name FROM Win32_Processor"]

      For more details on what is available, see the WMI and WQL
      documentation on MSDN (msdn.microsoft.com), and download the
      "WMI Code Creator" from Microsoft and browse the available
      namespaces, classes, and properties.


Modified Internal Commands:
---------------------------

    ACTIVATE - added some new options:

      /R(estore) - restore the original foreground window after
        updating the specified window.

      TRAY - move the specified window to the system tray.  Note: this
        is intended for 4NT and Take Command; no other console apps
      will work (Windows limitation); and GUI apps will only work
      if they have built-in tray support.

    ALIAS - now prevents a common user error when creating an @ and @@
      key alias with the same key.  (ALIAS will delete the previous
      alias definition so you won't have both active.)

    COPY - added new options:

      /F - don't create empty subdirectories (must be used with /S).

    COPY - now 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\NoFileFolderConnect
ion=0

    COPY - added support for copying streams attached to directory
      names.

    DEL - added support for deleting streams attached to directory names.

    DETACH - added the /Q(uiet) option to suppress displaying the
      process ID.
```

DIR - enhanced the /Q (show owner) option to display owners of remote files and folders.

DIR - enhanced the directory colorization. You can now use and/or /xor combinations (like IFF) with the attribute tests. (ColorDir does not support parentheses for grouping; the processing is left to right.) For example:

```
set colordir=com exe:bri red; rdonly .and. system:bri green
```

DO - added new options:

/C - each time through the loop, assign the next character in the expression to the argument; i.e.:

```
do var in /c ABC
    echo %var
enddo
```

will echo "A", "B", and "C".

FOR n SECONDS | MINUTES | HOURS - loops for the specified amount of time.

EVENTLOG - added a new option:

/Cn - set the Category. The value can be from 0-999999; Windows defines 0-7 as:

0 - None
1 - Devices
2 - Disk
3 - Printers
4 - Services
5 - Shell
6 - System
7 - Network

FFIND - added support for the /Sn syntax (to specify the maximum subdirectory recursion level).

FFIND (4NT) - now supports the "Editor" .INI directive.

FFIND - if you specify /U, FFIND will no longer print a CR/LF before the summary (so that it can be parsed by @EXECSTR).

FFIND - added new options:

/G - change to the directory where the file was found.

GLOBAL - added support for the /Sn syntax (to specify the maximum subdirectory recursion level).

GOSUB - now allows calling subroutines in other files. (This allows you to create files containing common subroutines.) The syntax is:

```
GOSUB "filename" label [args]
```

GOSUB - if you are in the batch debugger and attempt to call a nonexistent subroutine, GOSUB will now display a messagebox before exiting the batch file.

GOTO - if you are in the batch debugger and attempt to branch to a nonexistent label, GOTO will now display a messagebox before exiting the batch file.

HEAD - added support for displaying streams attached to directory names.

IF - added new tests:

IF PLUGIN modulename ... - returns TRUE if the specified plugin module is loaded.

IFTP - added some additional options:

/P0 - disable passive mode (overrides PassiveFTP directive)
/P1 - enable passive mode (overrides PassiveFTP directive)

KEYSTACK - added new options:

/A - append to existing KEYSTACK string. (I suspect this is going to be useless, but I've had a continuing stream of requests for it. If you actually find a use for it, please let me know!)

/R - read input from the specified file.

LIST - if you are displaying input from a pipe, the 'E' (invoke editor) key will copy the pipe contents to the clipboard and then open the editor. (You will need to do the paste into the editor manually.)

LIST - has some additional options:

(4NT) - added support for the 'E' (invoke editor) key (already in TC).

/C (4NT) - display the file in a separate screen buffer and restore the original buffer (& contents) upon exiting LIST.

/N - display line numbers on each line. You can also toggle line numbers with the 'L' key.

LIST - added support for displaying streams attached to directory names.

LOG - added a new option:

/A - (TCMD only) logs all output to the a log file (same name as the normal log but with a ".all" extension appended).

MKLNK - specifying /D for a filename with no additional links will no longer delete the file.

MOVE - added the /N options:

/NE - don't display errors
/NT - don't update JPSTREE.IDX

MOVE - now supports connected web folders. If an HTML file (i.e., with an .htm or .html extension) is moved, 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

OPTION - the "Registration" tab has a new button to remove the registration information from the computer.

OPTION - will now start at the last-referenced tab.

PDIR - enhanced the directory colorization (see DIR).

PLUGIN - added new switches:

/B - display the full pathname (only) for the specified module (also supports wildcards)

/P - pause after displaying a screen page

PLUGIN - the /I(nfo) switch now supports wildcards, and also displays the full pathname.

REN - added the /N options:

/NE - don't display errors
/NT - don't update JPSTREE.IDX

SETDOS - added option:

/X9 - enable / disable include lists

SHORTCUT - displaying a shortcut's values now shows the icon offset and the hotkey.

SHORTCUT - you can now define the hotkey for the link:

SHORTCUT "command" ["args" "dir" "description" "linkname" mode [iconfile iconoffset [hotkey]

The hotkey is in the same format as KEYSTACK; i.e. "Ctrl-Alt-[Shift-]x"

TAIL - added support for displaying streams attached to directory names.

TCTOOLBAR - you can now have up to 64 buttons. There are also some new options:

/C - clear toolbar

/R filename - load toolbar buttons from the specified file. (A /R will NOT clear an existing toolbar; you must use the /C flag in conjunction with /R for that.) The file is in the same format as the [Buttons] section in TCMD32.INI:

Bn=flags,text,command

Bn - The button number (1 – 32)
flags - 0=Echo, 1=Echo & Execute, 2=Execute w/o echo
text - The button text
command - The command line to execute.

TEE - added support for /A when TEE'ing to CLIP:

TEE - added new options:

/D - prefix each line with the date (in yyyy-mm-dd format)

/T - prefix each line with the time (in HH:MM:SS.MSS format)

TIMER - added the /Q(uiet) option.

TITLE - added a new option:

/P - expand PROMPT metacharacters in the title

TOUCH - /R now retains the exact time (100ns increments) of the source file.

TREE - added support for ranges (when used with /F).

TRUENAME - added support for junctions.

TYPE - added support for displaying streams attached to directory names.

New Debugger Features:

Batch debugger options (in BATCH.BCP) can now be set in the Debugger tab of the OPTION dialog.

You can change the line to be executed next when in debugging mode with either the "GOTO" dialog or by moving the caret to the line and either right clicking & selecting "Jump to This Line" or by pressing Ctrl-Shift-F11. Note that if you attempt to jump into or out of a DO loop or IFF block, bad things will happen!

A ^C / ^Break while debugging will now bring up the prompt:

Cancel batch job %s ? (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.)

```
Now supports saving Unicode files.

Added support for Ctrl-F1 (help for highlighted word) to Take
  Command.

The Alias window now uses the same syntax coloring as the debugger
  window.

Alt-F11 now invokes the Evaluate Expression dialog.

Ctrl-F9 invokes the Evaluate Expression dialog for the current
  line selection, or if there is no selection then for the current
  line in the debugger.

When running to a breakpoint, the debugger will no longer revert to
  single-stepping following a call to a compressed .BTM.

When running to a breakpoint, the debugger will highlight the line
  currently being executed.

The Find and Replace dialogs are now non-modal.  Also if you
  ve selected something on the line, the search field will be
  pre-populated with the selection.

Improved printer support.

Added support for background colors for keywords, text blocks,
  delimiters and numbers.

Added bracket and line highlighting.

Added case-fixing. When turned on, this will automagically fix
  keyword case.
```

13 Order Form

You can at any time apply your personal registration information 503 to a **trial** version in order to turn it into a **registered** version.

To order or upgrade any of our products, visit our web site and secure online store or email /fax us a copy of the form shown below. See Contacting JP Software 505 for our address and phone/fax numbers.

```
     JP Software                                  Standard Order Form
[2006/8]
     P.O. Box 328                                    email to
sales@jpsoft.com
     Chestertown, MD 21620 USA                        or fax to (410)
810-0026

     This order form covers our standard product line: new copies,
upgrades,
     our CD Suite, and manuals.

     For multi-system licenses, more shipping information, and answers
to
     other ordering questions, see http://jpsoft.com.  If you need
     additional information, or want to order directly from our web
site:

        For                            See
        ------------------------       ------------------------
        On-line order entry            http://jpsoft.com/
        Email orders / inquiries       sales@jpsoft.com
        Phone orders / inquiries       (410) 810-8818
        Fax orders                   (410) 810-0026
        US / international dealer list  http://jpsoft.com/


=====================================================================

     Name (first, last)
_____________________________________________

     Company
_____________________________________________________________________

     Address
_____________________________________________________________________


_____________________________________________________________________

     City, State/Prov, Post Code
_____________________________________

     Country
_____________________________________________________________________

     Phone _____________________________  Fax
_____________________

     E-Mail
_____________________________________________________________________
       (Will only be used by JP Software to notify you of upgrades or
        contact you with other information related to our products.)

     Register products to:  ___ Company     ___ Individual

     Where did you hear about our products?
_____________________________

=====================================================================
                                                          Total Price
     New Copies and Product Packs
     ___ 4NT ($74.95) ..............................  _____________
[NC120]
     ___ Take Command ($74.95) .....................  _____________
```

14 Copyright & Version

4NT 8.00 for Microsoft Windows 2000 / XP / 2003 / Vista
Take Command 8.00 for Microsoft Windows 2000 / XP / 2003 / Vista

Version 8.00 Help System

Language.dll translations by
French Christian Albaret
German Hans-Peter Grözinger

We gratefully acknowledge the contributions of Roger Byrne, Vincent Fatica and our other users.

This help material was last revised on Saturday, September 30, 2006

Index

- ! -

- $ -

- & -

- (-

- * -

- . -

- ? -

- @ -

- [-

- \ -

- ^ -

- _ -

- | -

- + -

- = -

- 4 -

- 6 -

- A -

- B -

- C -

- D -

- E -

- F -

- G -

- H -

- I -

- J -

- K -

- L -

- M -

- N -

- O -

- P -

- Q -

- R -

- S -

- T -

- U -

- V -

- W -

- X -

- Y -

- Z -