Personal Editor 32

Home Page: http://www.pe32.com

e-Mail: pe32@pe32.com

Contents

Personal Editor 32: MAIN FEATURES

Personal Editor 32 is a powerful ASCII file editor with a console interface.
It is the only real 32 bit upgrade of the Dos IBM Personal Editor and all its revisions called PE2 and PE3.

Personal Editor 32 (also called PE32) maintains the same user interface, the same profile file syntax and it can executes all the original editor commands.
It, however, uses a complete 32 bit core thus removing all DOS limitations, it adds many new powerful features like the multi language support, the syntax coloring and a lot of setting commands.

Personal Editor 32 is the best editor for editing large amount of data, for professional programmers and to simplify complex editing tasks.

Personal Editor 32 new release features are:

- NEW: Windows 10 compatibility!
- NEW: Full Python language integration!
- NEW: Improved editor control from VBScript and PowerShell!
- NEW: Windows 7 and 8 compatibility!
- NEW: Tabbed windows to change the current file using the mouse!
- NEW: Nested loops to repeat the execution of one or more commands!
- NEW: Context popup menu on the mouse right button click!
- NEW: IME input characters management (Double Byte Character Set)!
- NEW: Autocomplete for commands and filenames in the command line!
- NEW: Commands to count file lines, words and characters!
- NEW: The calc command can use and set numbers in the data file!
- NEW: Messages file to modify or translate all editor messages!
- NEW: Timed automatic save of all modified files!
- NEW: Export the syntax coloring in HTML and Rich Text Format!
- NEW: 9 improved syntax coloring files!
- NEW: 9 new national dictionaries for the spell checker!

Personal Editor 32 main features are:

- 200 editing commands and more than 40 setting commands.
- Different profile files for every file extension.
- Workspaces management to save and restore editing sessions.
- Codepages management for the international symbols.
- Menu to execute all PE32 commands and instructions.
- Open up to 1024 files in the same working session.
- The size of editable files is limited only by the amount of memory that Windows can allocate.
- Configurable syntax coloring for C/C++, Java, HTML, BASIC, ASM and .BAT.
- Autocomplete for all languages keywords and spelling checker using many national dictionaries.

- Pop-up commands history, recently used files and directories dialogs.
- Regular search expressions with substitution.
- Support for line marking, block marking and stream marking.
- Many block operations: move, copy, delete, overlay, fill and shift.
- Fully remappable keyboard and keystrokes macro available.
- Multiple ftp files editing and directories browsing.
- Send e-mail from the editor.
- Visual Basic Script commands and file execution inside PE32.
- Get and set PE32 attributes from Visual Basic Script commands.
- PE32.Py file to load Python procedures.
- PE32.VBS file to load Visual Basic Script procedures.
- Full 32-bit program, Windows XP/Vista/7/8/10 compatible.
- Fat, Fat32, NTFS, long file name and network path compatible.
- Timed autosave and backup copies on configurable files.
- Console interface with unlimited lines and columns.
- Bookmarks enhanced: saved, restored and a dialog to manage them.
- Sort all or a block of lines using different comparisons.
- Edit and save ascii files compressed in a zip archive.
- User defined synonyms to execute a sequence of commands.
- Open multiple files matching a wildcard specification.
- Save, load and run macro keystrokes.
- Creation a file containing all occurrences of a string.
- Commands to move the cursor simultaneously in all opened files.
- Command line and status line presence is configurable.
- Calc command to evaluate expressions.
- Encryption/Decryption password for each file.
- Window update is now two times faster.
- Search and replace simultaneously on all opened files.
- 26 bookmarks management: set, goto and remove commands.
- 255 levels undo and redo facility.
- Word wrap management, overstrike and insert typing modes.
- Restores the last cursor position on previously opened files.
- Block or word change case (upper, lower, invert and capitalize).
- Configurable format for date and time insertion instructions.
- Configurable context help on the current word.
- Instructions to divide the screen in multiple views.
- Instructions to open files in view mode (read only).
- Instructions to reformat lines and paragraph into margins.
- Full mouse management (wheel included).
- Clipboard and internal buffer copy, append and paste operations.
- Multiple files management and files merge and append options.
- Program execution without leaving the editor.
- Full file or block of lines printing on local or network printer.

Personal Editor 32 is completely customizable:

- Each keystroke can be assigned to one or more commands.
- Syntax coloring is modifiable and updatable by the customer.
- Different configurations (profiles) can be created and invoked.
- The editor can be adapted to the GUI style of another editor and can be integrated in the programming environments.
- Commands can be created for specialized editing tasks.

- All the languages, dictionaries and messages files can be modified and updated by the customer.

Personal Editor 32 interface

The Personal Editor 32 interface is divided into a file window plus a command line and a status line (the last two lines on the screen). The size of the file window is configurable with a command in the profile file.
The command and the status lines are always one line high but may be enabled or disabled.

On the command line you can type any commands described in the "COMMAND LINE INSTRUCTIONS" table followed by the Enter key.
If you want to execute more then a command in the command line include them between square brackets ([...][...]).

Additionally, you can type one or more commands described in the "PROFILE FILE INSTRUCTIONS" table if they are included between square brackets ([...][...]).

If you need to repeat the execution of one or more commands until a condition is valid you can include the commands between brace brackets ({[...][...]}). You can include loops in the main loop until 10 levels of nesting ({[...]{[...][...]}}).
Each loop have to include at least one conditional instruction in order to break it. The conditional instructions are:

- locate and change commands
- next and prev keyword/unknown/string/... commands
- tab and backtab word commands
- cursor movement commands

If no conditional command is included or the condition is always valid you can abort the loop pressing the Esc key.

Personal Editor 32: INSTALLATION AND USE

If you download the installation program called PE32.MSI from the web site, execute it and follow the simple instructions to install the files and to create a link to the program in the desktop. If you are a registered user please copy the license file in the installation directory.

If you have the PE32.ZIP file, unpack it in a temporary dir, then run the INSTALL.BAT program to create the dir C:\PE32 and to copy the distribution files and the license file there. You may also create a directory and copy the files manually.

During the installation step and at the first use of the visual basic script commands please verify that you are logged in Windows with the administrator rights or execute it using the option "Run as Administrator", otherwise some of the features cannot be enabled.

Create a link to the executable file (drag the c:\pe32\pe32.exe entry with the right mouse button down and drop it on the desktop then select the create shortcut to option).
If you want to execute the editor from the dos prompt without to specify the full path please add the installation directory to the system path variable (in the Control Panel - System dialog).
In Windows 7 and above, to remove all the window borders in the PE32 full screen mode, open the shortcut properties and select the Automatic Position of the Layout page (do it using the option "Run as Administrator").

To run the program use the syntax:

PE32 [switches] [commands] <pathname> [commands] ... <pathname> [commands]

where <pathname> is a specification of the file/s that you want to edit, [switches] are one or more of the optional switches here described and [commands] are one or more of optional commands here described.

It is possible to use long names, wildcard characters, and/or net names (e.g. PE32 \\server\mail\username\mailarchive.txt), but remember that if the file name contains spaces you have to enclose it between double quotes ("").
It is possible open files present in a zip archive if the pathname used ends with the “zip" extension or open files present in a ftp server if the pathname begins with the "ftp://" string.
If the pathname of the file to open contains a reference to an environment variable enclosed by '%' characters (e.g. %PE32%file.ext) the variable content is expanded in the pathname (e.g. C:\PE32\file.ext). This expansion is also present in all the editor commands regarding the files.

If the opened file extension is ".pe32" (the default workspace file extension) the editor restores the last state described in the workspace instead of opening the workspace itself. See the workspace command below for more details.

The optional [commands] parameters are a list of commands chosen from the "COMMAND LINE INSTRUCTIONS" or the "PROFILE FILE INSTRUCTIONS".

The "COMMAND LINE INSTRUCTIONS" must be enclosed in single quotes ('') or can be enclosed in square brackets ([]) while the "PROFILE FILE INSTRUCTIONS" must be enclosed in square brackets ([]).
In addition, to make a group of commands take as a single argument on the command line, it is best to enclose the group of commands in double quotes ("").
This is mandatory if the group of commands contain any delimiters (commas or spaces).

For example:

PE32 File1 "[top][cc] 'l/string/' [execute]"
Will open File1 positioned at the first occurrence of string.

PE32 File1 "[set display max]"
Will open File1 using the maximum display resolution.

The optional [switches] are:

- '/r' to open the preceding <pathname> in read only mode (e.g. PE32 file1 /r).
- '/?' to open the help file (e.g. PE32 /?)
- '/p<pathname>' to use the profile file with the specified pathname as default (e.g. PE32 /Pc:\pe32\pe32.pro)
- '/d' to do not clean the screen at PE32 exit (e.g. PE32 /d)

By default the PE32 profile file (PE32.PRO), the PE32 keywords files (PE32.KWD and PE32.KWD?), the PE32 license file (PE32.KEY), the PE32 Visual Basic Script procedures file (PE32.VBS), the PE32 Python procedures file (PE32.Py) and the PE32 message file (PE32.MSG) are read from the same directory as the executable file.

To install your preferred national language dictionary follow the instructions present in the "SYNTAX COLORING SETTINGS" paragraph.

The optional message file PE32.MSG, if present, is read to replace the original editor messages, the commands help strings and the all the menu items. Each message string or menu item has a unique number to identify it, as written in the PE32MSG.UK thatcontains the original english messages. To replace one or more editor messages create a PE32.MSG file an fill it with the modified lines copied from the original file.

The profile contains the personalization commands for PE32. If you want a personal profile, set the environment variable PE32PRO to the profile's pathname or use the /p parameter (e.g. SET PE32PRO=C:\PE32\MY.PRO).

See the "COMMAND LINE INSTRUCTIONS" and the "PROFILE FILE INSTRUCTIONS" tables for the instructions that can be used in this file.
Normally a profile file is divided in 4 different sections:

- a list of instructions to set the editor environment (set).
- a list of instructions to assign commands to the keys (def).
- a list of instructions to assign commands to the synonyms (syn).
- a optional list of instructions and commands to execute at profile load.

To register Personal Editor 32 as the editor executed by system when you double click on a ascii file you can use the "set openwith" command or you can manually register it as the default application to open text files (instead of Notepad) in the File types dialog in My computer, Options menu item. Remember that to open files with a long name containing spaces you have to enclose all parameters between double quotes (e.g.: C:\PE32\PE32.EXE "%1").

You may use the Personal Editor 32 in the Visual C++ environment by adding an item in the tool menu. Add a line in the Customize, Tools dialog and type PE32.EXE in the command field, type "$(FilePath)" "line $(CurLine)" "column $(CurCol)" "[key esc]" in the arguments item and type $(FileDir) in the initial directory item.

Invoking the tool will run PE32 on the currently selected file and place the cursor in the same position as the Visual C++ editor had it.

The mouse may be used while editing if you disable the Quick Edit Mode item in the Edit Options of the Options tab of the Dos window properties.
Reset this item in the PE32 shortcut properties too.
In Windows 10 you have to enable the item Use Legacy Console in the same properties dialog in order to use all the mouse functions.

The functions available with the mouse are the following:

1. Click on the left button to move the cursor to the indicated location.
2. Click on the right button to select a line or lines of text (like a-l or [mark line]) or to open the commands context menu if enabled (see the "set popupmenuon" command).
3. Drag with the left mouse button to select a block mark (like a-b or [mark block]).
4. Drag with the left mouse button while holding down the ctrl key to select a character mark (like a-c or [mark char]).
5. Drag with the left mouse button while holding down the shift key to select a line mark (like a-l or [mark line]).
6. Double click with the left button to select the word under the cursor (like c-w or [get word]).
7. Click on the left or right buttons, when the mouse cursor is on the status line and the scroll bar is moved (see "set statusline" ...). The left button changes the current file window by one page and the right button sets the new relative position in the current file window.
8. Using the mouse wheel to vertically scroll the file's lines. The mouse wheel management is available in only some operating systems and in only some window conditions (not in full window for example).
9. Within all dialog boxes you can use the mouse: with the left button click you can select menus and commands and with the double click you can execute the command.
10. If the tabbed windows are enabled and more than one file is opened you can click on the left button when the mouse cursor is on tab line (where the filename is displayed) to change the active file.

The arrow keys can be used with the shift, the control or the alt key pressed to continously mark all the characters under the cursor (see the "set markon" command).

When the Personal Editor 32 window takes focus (i.e. becomes the active window), it checks to see if any loaded files have been changed by other programs and if so it asks the user for permission to reload them.

Personal Editor 32: COMMAND LINE INSTRUCTIONS

All the following commands can be used on the invocation command line, the editors command line, or in a profile file.

Parameters that require one or more blanks must be enclosed in double quotes (e.g. edit "file name").
For the rest of this section, square brackets ([]) denote optional items and they are not to be typed. Also the < and > keys separate names for parameters and they are not to be typed either.

set display <rows> <columns>

Sets the number of display rows and columns. The maximum display rows is 512, and the maximum display columns is 1024.

The default values, if you don't set the display size in the profile file, are the current window rows and columns.

The rows and columns parameters can be substituted by the following fixed strings:
- "max": set the maximum possible rows and columns using the current font.
- "large font": alias for the standard setting 25 x 80.

"small font": alias for the standard setting 50 x 80.

set printer <printer name>

Sets the default printer port name for the print command.
You can use any LPTx port or a full net printer name (e.g. "\\server\laser printer") or use the word 'default' to select the system default printer.

The default value is ‘default‘.

set tabs <first col> <col step>
or
set tabs <col> ... <col>

When two parameters are used, PE32 sets the first tab stop column and the column steps to get to the subsequent tab stop columns.

When more than two parameters are used, PE32 sets a tab stop at each specified tab position.

The default value is 9 8.

set tabexpand <col>

Sets the number of columns used to expand tabs during the loading of a file or to compress blanks during the saving of a file. This value is independent of the previous set tabs values and it is used during the load or the save of a document only. Tabs when encountered in the input file are replaced with one or more blanks until the last blank is on a column that is a multiple of <col>.
The inverted process is followed during the save in the output file.

The default value is 8.

set margins <left margin> <right margin> <paragraph margin> <max column>

Sets the left and right column margins for the 'center in margins' and the'reformat' commands. The paragraph margin is used in the 'reformat' command only and is equal to the left margins if not defined.
If the max column parameter is present it sets the maximum number of columns in a line (the default value is 32767).
The default value is 1 80 1 32767.

set linetrim <off> <leading> <trailing> <compress>

Removes no blanks, leading blanks, trailing blanks, or multiple blanks from all file lines when the file is saved.
The default value is off.

set backup <file extensions or names including wildcard>

Creates a backup file called <filename>.BackupPE at every 'save' or 'file' command for all specified extensions or filenames (including wildcard)

The default value is no extension.

set colors <back> <fore> <mark back> <mark fore> <cmd back> <cmd fore> <dialog back> <dialog fore> <found fore> <found back>

Sets the background and the foreground colors for the normal file characters, the marked characters, the command line, all the dialogs and the found strings.

Valid colors are:
"black", "blue", "green", "aqua", "red", "purple", "yellow", "white", "gray", "lightblue", "lightgreen", "lightaqua", "lightred", "lightpurple", "lightyellow", "brightwhite".

The default value is Blue BrightWhite Red LightYellow Aqua Black Black Aqua BrightWhite Blue.

set syntaxcoloring<index> <back kw> <fore kw> <back op> <fore op> <back st> <fore st> <back no> <fore no> <back co> <fore co> <back sk> <fore sk>

Sets the background and the foreground colors for every kind of word(language keywords, operators, strings, numbers, remarks and systemkeywords) discovered during the syntax coloring process.
See previous colors list.

The index (none or a digit from 1 to 19) specifies which keywords fileto use (pe32.kwd or the file pe32.kwd<index>).

set syntaxcoloringext<index> <file extensions or names including wildcard>

Sets the file extensions (starting with a dot e.g. .cpp) or the filename (including wildcard) of the files to apply the specified syntax coloring.

The index (none or a digit from 1 to 19) specifies which keywords fileto use (pe32.kwd or the file pe32.kwd<index>).

set syntaxcoloringset<index> <set command, ...>

Executes different set commands for every file extension defined in the previous command. The set commands list must be separated with a comma and it can contain every set command unless itself.

The index (none or a digit from 1 to 19) specifies which keywords file to use (pe32.kwd or the file pe32.kwd<index>).

set syntaxcoloringpro<index> <profile file>

Load a different profile file for every file extension defined in the syntaxcoloringext command. The profile file can contain the only macros definition used by this extension while all other macros stay the same. To change the set commands please use the previous command.

The index (none or a digit from 1 to 19) specifies which keywords file to use (pe32.kwd or the file pe32.kwd<index>).

set historyon <scrolllockon/scrolllockoff/shift/control/alt/off>

Sets the key to press along with the up and down keys to show a dialog of previous commands. The key press only works when on the command line. From the history dialog you may select any recent command. That command is then brought back onto the command line for editing and execution.

The default value is scrolllockon.

set expandon <scrolllockon/scrolllockoff/shift/control/off>

Sets the key to press along with the space key to enable the synonyms auto expansion. It executes an expand command at the cursor position.
The auto expansion only works when the cursor is on the file window.

The default value is scrolllockon.

set autocompleteon <scrolllockon/scrolllockoff/shift/control/off>

Sets the key to press along with the space key to enable the words auto completion. It executes an autocomplete command at the cursor position.
The auto completion only works when the cursor is on the file window.

The default value is scrolllockon.

set markon <shift/control/alt/off>

Sets the key to press along with the arrow keys to enable the text marking.

The default value is shift.

set popupmenuon <rightbutton/leftbutton/doubleclick/off> <on/off>

Sets the mouse key to press to open the context popup menu.
The popup menu items change depending on the cursor position (command line or file lines) and on the features enabled at the mouse click.

The second parameter enable or disable the movement of the cursor at the mouse position before to open the context menu (in file space only).

The default value is rightbutton on.

set helpfile <pathname>

Sets the pathname of the help file used with the context help command on the current word. The program can open Windows Help files (.hlp) or Html Help files (.chm).

set confirm <on/off>

Enables or disables the confirm message during a 'save' or a 'file' command on an existing file.

The default value is off.

set insert <on/off>

Enables or disables the insert mode after the load of a file.

The default value is off.

set date <string>

Sets the format string used to display the current date. See the [date] command.

The format string is a sequence of characters and one or more of the following keywords (including the square brackets):

[d]	: the 1 digit days number
[dd]	: the 2 digits days number
[m]	: the 1 digit months number
[mm]	: the 2 digits months number
[yy]	: the 2 digits years number
[yyyy]	: the 4 digits years number
[month]	: the month name
[mmm]	: the first 3 characters of the month name
[d#]	: the number of the day of the week
[day]	: the day of the week
[ddd]	: the first 3 characters of the day of the week
[dn]	: the number of the day of the year

The default string is: [dd]/[mm]/[yy]

set time <string>

Sets the format string used to display the current time.
See the [time] command.

The format string is a sequence of characters and one or more of the following keywords (including the square brackets):
[24] : the 2 digits hours number in 24h
[12] : the 2 digits hours number in 12h
[mm] : the 2 digits minutes number
[ss] : the 2 digits seconds number
[am] : the am/pm string

The default string is: [24]:[mm]:[ss]

set encryption <on/off>

Enables or disables the encryption of the current file during the 'save' or 'file' commands.

The default value is off.

If encryption is on, you have to type the password and confirm it to encrypt the file. After that, the password stays valid for all following 'save' or 'file' commands or until another set encryption command is executed.

When encryption is active on a file, the keyword (Enc) follows the file name in the status line.

The 'set encryption on' command in the pe32.pro file activates file encryption on all files.

If the encryption is off and you open an encrypted file, you have to enter the password to decrypt the file and the encryption state for that file stays on. This password will be valid for all following 'save' or 'file' commands.

set restorecursor <on/off>

Enables or disables the initialization of the cursor to the last used position when loading any file in the most recently used file list.

The default value is on.

set statusline <on/off> <back> <fore> <modified> <on/off>

Enables or disables the status line and sets the background and foreground colors and the file modified colors.

The last parameter enables or disables the status line cursor that emulates a vertical scroll bar (show the current line position in relation of the file total number of lines). This relative position is indicated by an inverse video character mark. When this parameter is enabled you can click with the left mouse button on the status line to move to the next or to the previous page of the file by clicking before or after the highlighted character. Also with the right mouse button you can move the current line cursor to the selected relative
file position.

See previous colors list ('set colors' command).

When the status line is disabled all messages are displayed in the middle of the screen and will disappear after a key pressed.

The default values are on black brightwhite lightyellow on.

set commandline <on/off> <on/off> <on/off> <color>

Enables or disables the command line. When disabled the command line will disappear only when the cursor is on the data.

The second parameter enables or disables the restore of the cursor position in the active file when the up and down arrow keys are pressed while the cursor is on the command line.

The third parameter enables or disables the suggestion of the command you are typing in the command line. The most recent command or the most similar one are shown using the color defined as fourth parameter. If you want to autocomplete the suggested command hold down the left shift and control keys and release them. If this function is enabled the macro using the left control and shift keys in the command line are not executed.

The default value is on on on blue.

set oncommand <on/off>

Sets whether PE32 starts with or without the cursor on the command line when a file is opened.

The default value is on.

set onmenu <on/off>

Sets whether PE32 opens or doesn't open the command menu at load time.

The default value is off.

set bkspinfreespace <on/off>

Enables or disables the backspace key in free space. If disabled, a backspace in the free space moves the cursor onto the last non-blank character of the line.

The default value is on.

set eof <on/off>

Adds or suppresses the CTRL+Z byte at the end of the file.

The default value is off.

set linefeed <on/off>

Adds or suppresses the LF character after the CR character at the end of each line.

The default value is on.

set carriagereturn <on/off>

Adds or suppresses the CR character before the LF character at the end of each line.

The default value is on.

set wordwrap <on/off>

Enables or disables word wrap when the cursor reaches the right margin.

The default value is off.

set uppercase <on/off>

If on, PE32 forces to upper case all characters from a to z.

The default value is off.

set delimiters <characters...>

Sets the characters to delimit a word. The characters can be included between double quotes and can contain one or more of the following escape sequences:

\a = bell
\b = backspace
\f = form feed
\n = line feed
\r = carriage return
\t = horizontal tab
\v = vertical tab
\" = double quote
\\ = back slash
\<nr> = the ASCII code of the character to use

The default value is "\t\n\r ".

set blankcompress <on/off>

Enables or disables the compression of blanks to tabs when writing a file.

The default value is off.

set vbstimeout <milliseconds>

Sets the timeout for any Visual Basic Script commands and procedures.

If VBScript execution takes more then the specified milliseconds, a dialog box lets the user choose to continue or to abort the command.
Note that execution of the script continues while the dialog box is open and that the default timeout time is one minute (60000 ms). If continue is chosen then the dialog box will reappear again after the specified time expires again.

Any values equal or less to 0 disables the timeout.

See the paragraph "VISUAL BASIC SCRIPT USE" for more information.

set vbsrefresh <on/off>

Enable or disable the screen refresh during the execution of Visual Basic Script commands and procedures.

The default value is on.

See the paragraph "VISUAL BASIC SCRIPT USE" for more information.

set vbsload <pathname>

Load the specified script file after pe32.vbs load in order to add custom vbs procedures.

See the paragraph "VISUAL BASIC SCRIPT USE" for more information.

set pyload <pathname>

Load the specified Python file after pe32.py load in order to add custom Python procedures.

See the paragraph "PYTHON LANGUAGE USE" for more information.

set pyver <32/64>

Sets the version of the Python interpreter to use in the pyrun, pyexec and pycall commands.

See the paragraph "PYTHON LANGUAGE USE" for more information.

set wheelscrollline <lines>

Sets the number of lines to scroll using the mouse wheel. If the shift key is pressed the mouse wheel scrolls the file one page at time.

The default value is 1.

set codepage <number>

Changes the active codepage for the current console. The codepage number must be valid for the used operating system (Window 9x and ME don't support console codepages) and a true type font must be active.

The default value is the active console codepage.

set borders <characters>

Changes the characters used to draw menu and dialog borders. The character 'Í' used in the files top and bottom lines will change at the load of the next file.

The default value is "ÍºÄ³´¿ÀÁÂÃÙÚ".

set cursor <size> <size>

Changes the cursor replace and insert sizes. The sizes can range between 1 (a thin line) and 100 (a complete character).

The default values are 20 and 90.

set showbookmarks <on/off>

Shows or no the bookmarks position in the current file.

The default value is on.

set locateswitch [*-mnewsaxofichtbr#lg]

Sets the default switches used in the locate command. For every switch present in this command the relative locate switch is enabled by default and the presence of the switch in the locate command disable it.

The default value is no switch active.

set changeswitch [*-mnewpaxoichtb#]

Sets the default switches used in the change command. For every switch present in this command the relative change switch is enabled by default and the presence of the switch in the change command disable it.

The default value is no switch active.

set linesback <value>

Set the number of lines to search back for unknown words in the autocomplete command. A value equal or lower to 0 disable the search.

The default value is 10000 lines.

set linesforward <value>

Set the number of lines to search forward for unknown words in the autocomplete command. A value equal or lower to 0 disable the search.

The default value is 10000 lines.

set tabbed <on/off> <columns>

Enables or disable the tabbed windows if more than one file is opened and sets the tab width where the filename is displayed.
If the tabbed windows are enabled you can change the current file using the mouse with a left click on the filename.

The default value is off and 14.

set autosave <value>

Set the number of seconds after the editor start or after the last autosave to automatically save all modified files.
A value equal or lower to 0 disable the autosave.

The default value is 0 (no autosave).

set charset <oem/ansi>

Set the active codepage used to display all the characters.
If this command is not executed the set codepage command is used to set the active codepage. If this command is executed with the oem parameter the system oem codepage is activated. If this command is executed with the ansi parameter the system ansi codepage is activated.

The default value is oem.

set tabinsert <on/off>

Enable or disable the lines indent pressing the tab key in insert mode.

The default value is off.

set openwith <extensions...>

Set PE32 as the default explorer application used to open a file with an extension included in the list.

The default value is no extension.

set currentline <on/off/toggle>

Highlight or not the current line of the active file. The toggle option invert the state of the command.

The default value is off.

set currentpos <on/off/toggle>

Highlight or not the current cursor position in the active file when the cursor is in the command line. The toggle option invert the state of the command.

The default value is on.

set pecompatibility <on/off>

Enables or disables the original PE compatibility mode. If disabled PE32 works exactly as now, then the behaviour of some commands is not completely the same of the original PE program.

When the compatibility mode is enabled the following commands change the behaviour as described below:

page up/down: the page is scrolled with an overlap of one line
page up/down all: the page is scrolled with an overlap of one line
first nonblank: move at the end of a line with blanks only
file/save ... tabs: after a ' or " character the blanks compression ends
dos ...: it acts in the same console window as the "system" command and it removes the optional ' ' after the command
save <name>: it doesn't change the active file

The default value is off.

set abbrev <on/off>

Use or no the commands abbreviated form in the help and .keydefs commands.

The default value is off.

dir [<pathname>] <filters...>

Creates a dialog box with the list of all files present in the pathname specified and with the filtered name and extension.

In the dialog box the you choose a file using arrows keys, edit it with the Enter key, abandon the dialog with the Esc key, select a file by typing the first letter, edit multiple files using the numpad plus key or edit all files using the numpad multiply key.

By pressing the insert key, the dialog's information is copied into a file called .dir and by pressing CTRL+C on a file, the filename is copied into the clipboard.

By pressing the Enter key on a directory name, the directory files are listed, while using the numpad minus key on a directory name, it will be the current directory at the dialog exit.

If the pathname is related to a ftp file (e.g.: ftp://ftp.abcdef.com) a connection to the ftp site is opened and all the files present in the directory that matches the pathname are shown.

With the mouse click you can select files and directories and with the double click you can open a file or a dir.

erase <file>

Removes the file specified.

rename <oldfile> <newfile>

Renames the not opened file <oldfile> as <newfile>.

name <file>

Renames the current opened file to the specified name.

macro <profile pathname>

Activates the specified profile file.

All existing key settings are preserved unless overwritten by commands in the loaded profile file.

cd [<path>] or chdir [<path>]

Changes the current directory to the path specified or displays the current directory if no path is present.

line <row> <column>

Moves the cursor to the line number and column number specified.
This command also works with only a line number.

column <number>

Moves the cursor to the column number specified.

pe32 [/s] <file> [commands] ... <file> [commands]

Executes another copy of PE32, within the current one, that opens the specified files and executes the specified commands.

If the first optional /s parameter is not present the execution of the current PE32 process continue.
If the /s first parameter is present the current PE32 process is suspended until the new PE32 process ends.
The only way to exchange data between the PE32 processes are the clipboard or the file.

Include the optional files and the commands between double quotes ("") if they contain spaces.

edit [<filename>] [mri]

Opens the file specified in a new file window. If the m parameter is added, PE32 inserts the file into the currently opened file at the current cursor position.

If filename is ".keydefs" PE32 creates a file containing all key macro assignments and current set commands.

If the filename is "-", the previously opened file is activated.
If the filename is not present, the next opened file is activated.
If the r parameter is added, the file is opened in read only mode.
If the i parameter is added, the workspace file is opened instead of being restored.
If wild card characters are present in the filename all matching files are opened.

If the filename contains a reference to an environment variable enclosed by '%' characters the variable content is expanded in the filename.

If the file extension is ".zip" all files compressed in the archive are expanded and opened. After the editing if you save one or all the files expanded, these files are compressed in the original archive.

If the file is located in a ftp site (e.g.: ftp://ftp.abcdef.com/file) a connection to the ftp site is opened and the file is downloaded.
At every file save or when the file is closed it is uploaded to the ftp site and the connection is closed.
If multiple files are opened from the same ftp site the user name and the password asked for the connection are saved and the connection stays opened until the last file is closed.

view [<filename>]

Opens the file specified in a new file window in read only mode.
If the filename is "-" PE32 re-opens the previously opened file in read only mode.
If the filename is not present, PE32 re-opens the next opened file in read only mode.

If wild card characters are present in the filename, all matching files are opened in read only mode.

If the filename contains a reference to an environment variable enclosed by '%' characters the variable content is expanded in the filename.

file [<filename>] [tabs/notabs]

Saves the current file with the optional name specified and closes the file window.
The tabs and notabs switches can be used after the filename only and they are used to compress blanks to tabs or not.
If no switch is present the compression will follow the set blankcompress command.

If the filename parameter is "*" all opened files are saved with their current names and the program is closed.

If the filename contains a reference to an environment variable enclosed by '%' characters the variable content is expanded in the filename.

save [<filename>] [tabs/notabs]

Saves the current file with the optional name specified without closing the file window. The tabs and notabs switches can be used after the filename only and they are used to compress blanks to tabs or not. If no switch is present the compression will follow the set blankcompress command.

If the filename parameter is "*" all opened files are saved with their current names.

If the filename contains a reference to an environment variable enclosed by '%' characters the variable content is expanded in the filename.

append <filename>

Append the marked text to the specified file. If the file doesn't exist it is created.

zip <zipfilename>

Compress the current file into a zip archive with the name specified.
The current file must not be opened from a zip archive and it stays opened after the command execution. If the zip archive is already present the command adds the file to the archive.

unzip <filename>

Extract the current zipped file into an ascii file with the name specified. The current file has to be opened from a zip archive and it stays zipped and opened after the execution of the command.

quit [all|*] [y]

Abandons all modifications on the current file and closes the file window.

If the all or the * parameter is added, PE32 repeats the quit command for all opened files.

If the y parameter is added, PE32 doesn't ask for confirmation if the file is modified.

print

Prints the whole file on the default printer.

printdoc [m]

Prints the whole file, or the marked lines if the 'm' parameter is present, using custom printer and page settings.

change/oldstring/newstring/[*-mnewpaxoichtb#]

Searches (from the current cursor position up to the end of the file) for an occurrence of "oldstring" to be replaced with "newstring".

The character used to delimit the strings is the first one non space following the "change" or the "c" word.

If the string is not found the current macro (if present) is aborted unless the 'i' parameter is present.

Optional parameters are:

*	repeat the replacement until the end of the file.
-	start the search from cursor position backwards to the beginning of the file.
m	only replace marked strings.
n	only replace not marked strings.
e	search for a case sensitive string.
w	search for a whole word only (see the set delimiters command above).
p	select the matched text with a mark and prompt for a confirmation. You can press Y to confirm the change and continue, press N to ignore the change but continue, press A to change all subsequent matches without further confirmation, or press ESC to abort the command at the current position.
a	repeat the replace command on all opened files.
o	start the replace command from the top of the file (or from the bottom if the '-' parameter is present), then ignoring the cursor position.
i	suppress aborting the current macro if the string is not found.
c	come back in the command line if the string is found.
h	show the found string in the center of the screen.
t	show the found string at the top of the screen.
b	show the found string at the bottom of the screen.
#	show the number of changed strings.
x	enable the use of searching expressions in the search string (oldstring) and in the replace string (newstring).

Valid searching expressions in the search string are:

?	Matches any single character (e.g. l/a?b/x search for an 'a' followed by any single character and a 'b').
*	Matches zero or more occurrences of the previous character (e.g. l/a *b/x search for an 'a' followed by zero or more blanks and a 'b').
+	Matches one or more occurrences of the previous character (e.g. l/a +b/x search for an 'a' followed by one or more blanks and a 'b').
[c1-c2]	Matches any character within the class c1-c2. All characters starting with the character before the dash (-) and ending with the character after the dash will be included in the search (e.g. l/a[c-e]b/x search for an 'a' followed by a 'c' or a 'd' or an 'e' and a 'b').
[c1...cn]	If none of the supplied characters are dash (-) then PE32 matches any of the specified characters (e.g. l/th[aeiou][ts]/x from here will match THAT, THIS, parenTHESis).

The above two forms can be mixed (see :a below).

[^c1-c2]	Matches any character not in the class c1-c2 (e.g. l/a[^c-e]b/x search for an 'a' followed by a character that is neither 'c' nor 'd' nor 'e' and a 'b').
[^c1...cn]	If none of the supplied characters are dash (-) then PE32 matches anything but the specified characters (e.g. l/th[aeiou][^ t]/x from here will match oTHERs).

The above two forms can be mixed (see :a below).

^ Anchors the following match at the beginning of the line (e.g. l/^an/x search for an 'a' followed by a 'n' at the beginning of the line).

$ The match succeeds only if at the end of the line. The '$' sign must be at the end of the searching string (e.g. l/an$/x search for an 'a' followed by a 'n' at the end of the line).

| Matches the string previous to the '|' or the string after the '|' (e.g. l/ab|ac/x searches for an 'a' followed by a 'b' or an 'a' followed by a 'c').

\ Takes the next character literally and not as a searching expression. If the next character after the backslash is a 'c', a digit, or a parenthesis, see the below expressions.
If the next character after the backslash one of thefollowing the relative character is replaced:
\bbackspace
\nnew line
\rcarriage return
\ttabulation
\fform feed
\Eescape
(e.g. l/a\+b/x search for an 'a' followed by a '+' and a 'b').

\c Places the cursor at that position if the match is successful (e.g. l/ab\cc/x search for an 'a' followed by a 'b' and a 'c' and places the cursor on the 'c').

\(a\) Tags the pattern <a>. This tag can be referred to in subsequent parts of the search string or in the replace string.
(e.g. l/\(abc\)[0-9]\1/x search for 'abc', any digit, then 'abc'. Note that "\1" refers to the tagged pattern 1, which is "abc")

\n Refers to the tagged pattern <n>. N is a number from 1 to 9.

{} Groups an expression. For example to search for one or more occurrences of "abc" use "{abc}+".

:a Matches all alphanumeric characters. Is the equivalent of "[a-zA-Z0-9]".

:b Matches all white space characters. Is the equivalent of "[\t]+".

:c Matches all alphabetic characters. Is the equivalent of "[a-zA-Z]".

:d Matches all numeric characters. Is the equivalent of "[0-9]".

:f Matches all filenames. Is the equivalent of "[^\\[\]:<>|=+;,]+".

:h Matches all hex numbers. Is the equivalent of "[0-9a-fA-F]+".

:I Matches all integers. Is the equivalent of "[0-9]+".

:l Matches all valid C identifiers. Is the equivalent of "[a-zA-Z_][a-zA-Z_0-9]+".

:w Matches all words. Is the equivalent of "[a-zA-Z]+".

Note: To search for a colon, you must prefix it with a backslash (ie - \:).

Valid searching expressions in the replace string are:

& The entire search string will be substituted at that point. For example if the search string is "abc" and the replace string is "12&34", then "abc" will be replaced with "12abc34".

\ Takes the next character literally and not as a searching expression. If the next character after the backslash is a digit see the below expression.
If the next character after the backslash one of the following the relative character is replaced:
\bbackspace

\nnew line
\rcarriage return
\ttabulation
\fform feed
\Eescape
(e.g. c/abc/\n/x replace all 'abc' strings with a new line character).
\n: Refers to the tagged pattern <n> in the search string. N is a number from 1 to 9.
(e.g. c/\(dog\) = \(cat\)/\2 = \1/x replace every occurrence of "dog = cat" with "cat = dog").

locate/searchstring/[*-mnewsaxofichtbr#lg]

Searches from the current cursor position to the end of the file for "searchstring".

The character used to delimit the strings is the first one non space following the "locate" or the "l" command.

The "locate" or the "l" commands may be omitted if the delimiter character is a slash "/".

The last delimiter character may be omitted if no parameter is present.

If the string is not found the current macro (if present) is aborted unless "i" parameter is present.

Optional parameters are:

* repeat the search up to the last occurrence of the string.
s select the matched text with a mark.
f repeat the search until the end of the file (or files) is found and copy all lines containing the matched text into a new file called .found.
sf with both "s" and "f" switches repeat the search until the end of the file (or files) is found and copy the matched text only into a new file called .found.
r repeat the search until the end of the file (or files) is found and move all the lines containing the matched text into a new file called .found.
repeat the search up to the last occurrence and show the number of found strings.
l repeat the search until the end of the file (or files) is found and fill a list with all found lines. From the list you can choose a line and goto on it with the mouse or the enter key.
g highlight the found strings with the colors defined in the set colors command.

See the change command for the meaning of all other optional parameters and for the meaning of the searching expressions in in the search string.

Examples of search string:

abc Matches 'abc'.

^$ Matches a line with no characters. (the beginning of the line followed by the end of the line)

^abc Matches a line which starts with 'abc'.

^abc$ Matches a line which contains only 'abc'.

[A-Z][0-9] Matches a capital letter followed by a digit.

[^A-Za-z0-9] Matches a character which is not alphanumeric.

be?t Matches 'be', followed by any character, followed by a 't'. (ie - bert, belt, bent).

ab*c Matches 'ab*c'. Note that "*" tells the pattern matcher to take the asterisk literally.

{abc}+ Searches for one or more occurences of "abc".

[A-Z][A-Z]*[.?,!] Matches a capitalized word.

if|else Matches the word 'if' or the word 'else'.

\([A-Z]+\)?*\1 Matches a line which contains two or more occurences of the same upper-case word. Note that "\1" refers to whatever [A-Z]+ matched.

def <key> = [command]...[command]
or
define <key> = [command]...[command]

Defines a key as a sequence of commands.

The keys can be normal (name), shift (s-name), control (c-name), alt (a-name) or control+shift (u-name).
The list of available keys name is present at the end of this document (see the Personal Editor 32 keys name table).

Commands that may be used after the equal sign are listed in this section ("COMMAND LINE INSTRUCTIONS") and in the following section ("PROFILE FILE INSTRUCTIONS") and have to be enclosed between square brackets ([]).

If a command contains square brackets you have to enclose it in double quotes ("") (e.g. def a-a=["locate/[/"]).
If you want to redefine one or more keys in a def command you have to enclose each in double quotes (e.g. def a-a=["def a-b=[up]"]).
If, between double quotes, you need to insert a double quote you have to precede it with a back slash
(e.g. def a-a=["def a-b=[\"set date [yy]/[mm]/[dd]\"]"]).

If you want to use a set command in a def command you have to enclose it in square brackets (e.g. def a-a=[set display 25 80]).
One or more of the commands could be a previously defined synonym.
Constant strings must be enclosed in single quotes ('').

Note: that one key may invoke the actions of another key by referencing it in the sequence of commands by using the key command (e.g. [key enter <nr>] or [key u-1 <nr>]). The optional parameter <nr> is the number of times to repeat the key (or the synonym).

Remarks can be entered between commands using the syntax [* remark].

A long list of commands can be divided into several lines by ending each line (except the last one) with a back slash.

If the definition is typed on the command line it will be valid for only the current working session. If saved into the profile file, however, it will be valid whenever the profile is loaded.

To remove a previous defined keys sequence you can execute it with an empty command list (e.g. def a-a=).

syn <name> = [command]...[command]

Defines a command name to use as a synonym for a sequence of commands.

The name can be up to 32 characters long and it must be different from any existing command.

The syntax is identical to the rules for the def command. The difference between syn and def is that def assigns the sequence to a key while syn assigns the sequence to a new keyword.

One or more of the commands could be a previously defined synonym.

key <key> [<nr>]

Executes all commands assigned to the key <nr> times. <nr> defaults to one time.

about

Displays the "about box" with release and license information.

ver

Displays the "about box" with release and license information.

system <command> <parameters> <...>

Executes a shell command in the same console window.

dos <command> <parameters> <...>

Executes a shell command in a different console window.
To execute a dos command in a different window you can type:
dos "command.com /c dir *.txt" or
dos "command.com /k dir *.txt"

mail <address> "object" [attachment]

Sends an e-mail to the address and with the object specified, using the current edit file as message body and attaching an optional file to the e-mail.

history

Opens the command history dialog box.
In the dialog box you may choose a command using arrows keys, run a command with the Enter key, abandon the dialog with the Esc key, or delete a command from the history using the delete key.
If the command line is not blank when the dialog is opened then only commands that start with what is on the command line will be displayed.

With the mouse click you can select a command and with the double click you can copy the command in the command line.

mru

Opens the most recently used files dialog box.

In the dialog box you may choose a file using arrows keys, edit a file with the Enter key, abandon the dialog with the Esc key, edit multiple filesusing the numpad plus key, edit all files using the numpad multiply key, or delete a file from the most recently used list using the delete key.

With the mouse click you can select a file and with the double click you can open it.

help [<set command> or <key name> or <synonym command>]

Help followed by one of the set commands (e.g. help display) or one of the key names (e.g. help s-f1) or one of the synonym commands (e.g. help if) fills the command line with the related command with all of its current parameters.

Help followed by the "char" keyword writes the ASCII code of the character under the cursor in the command line as [xxx].

With no parameters, help opens the commands help dialog box.

In the dialog box you may choose a command using arrows keys, run a command with the Enter key, abandon the dialog with the Esc key, or move to a command by typing it's first letter.

With the mouse click you can select a command and with the double click you can copy the command in the command line.

[<3 digits number>]

Convert the 3 digits number in the relative ascii character and write it at the cursor position.

vbsexec <Visual Basic Script commands>

Executes one or more Visual Basic Script commands. If you want to execute more than one command separate them with colons (:).

Without any vbs commands the function displays the list of already loaded vbs functions and lets you to choose which function to execute.
If the function needs one or more parameters, they are requested.
See the paragraph "VISUAL BASIC SCRIPT USE" for more information.

vbsrun <filename>

Loads and parses a file containing Visual Basic Script commands and procedures. This makes any included functions available for execution by the vbsexec command. Without the filename parameter the function displays the list of *.vbs files present in the current directory and lets you to choose which file to execute.

See the paragraph "VISUAL BASIC SCRIPT USE" for more information.

pyexec <Python command>

Executes one Python command, the command string is case sensitive.

See the paragraph "PYTHON LANGUAGE USE" for more information.

pycall <Python function>

Call a Python function with zero or more parameters.
The function name is case sensitive.
Without a Python function the command displays the list of already loaded Python functions and lets you to choose which function to call.
If the function needs one or more parameters, they are requested.

See the paragraph "PYTHON LANGUAGE USE" for more information.

pyrun <filename>

Loads and parses a file containing Python commands and procedures. This makes any included functions available for execution by the pycall command. Without the filename parameter the function displays the list of *.py files present in the current directory and lets you to choose which file to execute.

See the paragraph "PYTHON LANGUAGE USE" for more information.

calc <expression>[=$row,col]

Calculates the result of the expression and displays it in the status line in decimal and hexadecimal format if the equal sign at the end of the expression is not present. If the equal sign is present at the end of the expression the result is written after it. If after the equal sign the $row,col syntax is used, the result is written in the current file at the specified row and column (if one or both of the row and col parameters are omitted the current cursor position is used).
If the expression parameter is not present it looks at the marked text in the current file as an expression.
If the only parameter is one of these: +, -, * or / it executes this operation on all numbers present in the marked text.

All numbers may be expressed in decimal or hexadecimal format if preceded by 0x.

The operators available are:

+, -, *, /, ()	arithmetical operator and parenthesis
SIN, COS, TG, CTG	sine, cosine, tangent and cotangent
ASIN, ACOS, ATG	arc sine, arc cosine and arc tangent

EXP, SQRT, LOG: exponential, square root and logarithm

INT	integer value
|	absolute value
^	power of
<, >, !	logical operator (less, greater, and inverse) $row,col : the value present in the current file at row or $(row,col) and column is used in the evaluation (if the or $(+row,+col)row or the column parameters are omitted the or $(-row,-col)current cursor position is used). The parenthesis are mandatory if the current row or column increment ordecrement (+row or -row) is present.

Any word not included in the previous operators list is considered as a variable. If the variable has no value a dialog is shown to enter it, otherwise it's value is used in the expression.

sort [column] [column] [dinrck]

Sort all file lines at the columns specified in the parameters.
Without parameters it sorts ascending all file lines starting from the first column. If you specify one or two columns the sort is done looking at the characters included between the two columns (or from the first column to the end of the line if the second column is not present). If a mark is present only lines belonging to the mark are sorted. If a block mark is present and there is no column parameter the comparison use the marked characters only.
If the first column value is negative or the last parameter contains a 'd' the sort is descending. If the last parameter contains an 'i' the comparison ignore the character case. If the last parameter contains a 'n' the comparison is done converting the text into a number. If the last parameter contains a 'r' the comparison is done removing the leading blanks.
If the last parameter contains a 'c' the sort is done removing all the duplicated lines (compact).
If the last parameter contains a 'k' the sort is done removing all the lines with the same marked or specified columns.

menu

Opens a menu to execute all PE32 commands and instructions.
With the mouse click you can select menus and commands and with the double click you can execute a command.

load <filename>

Loads the current macro from the filename. The current macro can be executed by using the [macro play] command (assigned to c-g in the default profile).

run <filename>

Loads and executes the current macro from the filename.

Macros recorded with the [macro define] command (and saved with the write command) must act entirely on one file and entirely in cursor data mode. The keystrokes typed in this process are converted to profile commands and saved into the specified filename.

The run command, however, can run files containing any profile commands.

Thus after automatically recording keystrokes and saving them as a macro to a file you may edit that file and manually add any additional commands. This allows you to easily operate on multiple files and to switch from edit mode to command mode as needed.

write <filename>

Writes the current macro into the filename.

The current macro is defined by using the [macro define] command (assigned to c-q in the default profile). When invoked, [macro define] saves the actual commands that result from each keystroke as they are typed. A second [macro define] will close the definition of the macro.
During recording of a macro you can pause and resume recording using [macro pause] command (assigned to a-q in the default profile). Macro recording may be ended while it is paused.
Once the macro has been defined it can be executed by using the [macro play] command (assigned to c-g in the default profile).
Macros must act entirely on one file (file switching using the edit command is ignored) but may be played on numerous files.

demo <fast/slow/step/end>

"Demo Fast"	refreshes the screen after each command included in a macro.
"Demo Slow"	also adds a 250ms delay between each command.
"Demo Step"	isplays on the status line the last macro command executed and waits for a key before continuing. The ESC key will abort the execution of the macro. Any other key will step into the command.

These commands allow you to visually debug the macros.

"Demo End" stops the demo mode.

open

Creates a dialog box with the list of all opened files.

In the dialog box you may choose a file using arrows keys, open it with the Enter key, abandon the dialog with the Esc key, select a file by typing the first letter, quit a file using the numpad minus key, save a file using the numpad plus key, save all files using the numpad multiply key or quit all files using the numpad divide key.

With the mouse click you can select a file and with the double click you can open it.

workspace <filename>

Creates a workspace file called <filename> with the extension ".pe32" and activates it as the current workspace.

The editor creates saves all the PE32 commands needed to restore the actual editing environment (the opened files, their cursor positions and the active mark) into the workspace file.

Once the workspace file has been saved, it can be restored and reactivated by opening it as a file with Personal Editor (in this case you must specify the .pe32 extension). This causes the instructions in the file to be executed, restoring the current editing state.

An active workspace is updated only with the "quit all" command.

In order to see or modify a workspace file you can add the ignore (i) flag at the end of the "edit <file>" command.

If a workspace is active and you open another workspace the previous workspace is saved before the new workspace is opened.

If the workspace command has no parameters or the parameter is a directory name (ending with backslash) a dialog is opened to show all the workspaces present in the current or in the specified directory.

bookmarks

Opens a dialog box to show all bookmarks present in the current file.
For each of them you can see the line and column numbers and the first part of the file line where the bookmark was set.
All bookmarks are saved at the file quit and restored at the file load.

In the dialog box you may choose a bookmark using arrows keys, go to the bookmark with the Enter key, abandon the dialog with the Esc key, set the bookmark with the numpad Plus key, remove the bookmark with the numpad Delete (or Minus) key, remove all bookmarks with the divide numpad key or move the cursor to a bookmark by typing it's letter.

With the mouse click you can select a bookmark and with the double click you go to the bookmark position.

html [<filename>]

Convert the current opened file in a new file with the .html extension and the html format.
The original data format and the foreground colors are converted.
If the filename is not present the .html extension is appended to the current filename.
If the filename already exists it is opened and the content is erased before the conversion (the disk data are overwritten at the first save operation).

rtf [<filename>]

Convert the current opened file in a new file with the .rtf extension and the rich text format.

The original data format and the foreground colors are converted. If the filename is not present the .rtf extension is appended to the current filename.
If the filename already exists it is opened and the content is erased before the conversion (the disk data are overwritten at the first save operation).

lines

If is present a mark in the current file this command shows the number of lines present in the mark else it shows the number of lines in the current file.

chars

If is present a mark in the current file this command shows the number of characters present in the mark else it shows the number of characters in the current file (carriage return and line feed are ignored).

words

If is present a mark in the current file this command shows the number of words present in the mark else it shows the number of words in the current file.
See the set delimiters command to define or change the word delimiters.

import <filename>

Loads all data present in the specified file and imports it into the current file being edited.

With this command you can insert the text from <filename> into a macro or into a synonym. This allows you to customize your macros (for instance with different behaviors for different users) without having to have completely separate profiles.

This also allows you to create file skeletons, for example program headers, that will have specific information filled in when they are imported.

All characters are imported as ascii code (as 'char' into the current file) and only commands included between '<% ... %>' are interpreted as PE32 commands. Thus PE32 commands outside '<% ... %>' are just added to the file as text but PE32 commands inside '<% ... %>' have '<%' and '%>' ignored while the commands are run (interpreted).

Any CRLF codes are translated into an [insert line] command unless they are preceded by 2 back slash characters ("\\").

Using these substitution functions you may input words during macro or synonym execution, and execute conditional PE32 commands.

The syntax of these substitution commands is:

<%@Command String_Variable_Name=[len]Default_String_Value%>

where the @Command can be one of the following:
no command before the string name

@input
@set
@calc
@if
@else
@endif
[pe command]

and the String_Variable_Name is the name of the variable to be assigned with the Default_String_Value or the operator prompted value.

The Default_String_Value can be a fixed value or the name of a previously assigned String_Variable_Name.

The String_Variable_Name can be globally or locally defined:

- . A global variable must begin with the '_' character and is initialized only once.
- . A local variable must not begin with the '_' character and is initialized every time the macro containing a usage of the variable starts.

For first three forms (no command, @input, and @set):

The optional [len] parameter contains the minimum length of the string to write into the file being edited. If the string is shorter than this length then it ispadded with blanks. The string is aligned to the right if [len] is positive and to the left if it is negative. If the string is longer than the supplied length (or no length is specified) then the full string is substituted.

No command :

Used to enter a string variable and copy it to the current macro.
String_Variable_Name is optional if the Default_String_Value is present.
For readability the String_Variable_Name can be included between double quotes.
Default_String_Value is optional if the String_Variable_Name is present.
For readability the Default_String_Value can be included between double quotes.

@input :

Used to enter a string variable (already assigned or not).
String_Variable_Name is optional if the Default_String_Value is present.
For readability the String_Variable_Name can be included between double quotes.
Default_String_Value is optional if the String_Variable_Name is present.
For readability the Default_String_Value can be included between double quotes. If the Default_String_Value is not present and the variable is already assigned the actual value is used.

@set :

Used to set a string variable with a fixed value.
String_Variable_Name is optional if the Default_String_Value is present.

For readability the String_Variable_Name can be included between double quotes.
Default_String_Value is optional if the String_Variable_Name is present.
For readability the Default_String_Value can be included between double quotes.

@calc :

Used to evaluate an expression and set the string variable value.
String_Variable_Name and Default_String_Value are mandatory.
Both can be included between double quotes.
The optional [len] parameter contains the format of the expression value to write into the macro. The [len] integer part is the minimum number of digits to write, while the decimal part is the minimum number of digits to write to the right of the decimal point.

@if :

Used to test two string variables or one variable and a string constant.
String_Variable_Name is mandatory.

Default_String_Value is mandatory and may be a previously defined string.
The operators may be one of the following:
'=' : string is identical to
'<>' : string is different from
When both strings can be converted to numbers then additional operators
are allowed:
'<' : number is less than
'>' : number is greater than
'<=' : number is less than or equal to
'>=' : number greater than or equal to
The if/else/endif commands cannot be nested.

@else :
@endif :

Used to close the @if branches.
They have no parameters.

[pe command]

Used to execute one or more PE32 commands.
If any [pe commands] are present no other substitutions or string variables may be present. That is, there is no nesting of the feature allowed.

When a string variable is encountered in a macro for the first time, the actual value is prompted for from the user. On subsequent occurrences of the same string variable in the same macro, the first value is automatically expanded into the macro code again.

e.g.: <%"Enter your name"="My name"%>
Fetches actual user name and places it into both the local variable "Enter your name" and into the current file.
If the user just hits enter then "My name" is used.

<%=[10]My first name%>
Fetches actual user name and pads it with blanks on the left until it is at least 10 characters long and places it into both the local variable "My first name" and into the current file.

<%_Type your name=[-20]%>
Fetches actual user name and pads it with blanks on the right until it is at least 20 characters long and places it into both the global variable "_Type your name" and into the current file.

<%@input _Type your name=[-20]%>
Fetches actual user name and left justifies it if it is shorter than 20 characters and places it into the global variable "_Type your name".

<%@set Type your name="John"%>

Set the value of the local variable "Type your name" to "John".

<%@calc number=[2.0]"10*(var1+var2)/2"%>
Evaluate the expression (asking the var1 and var2 values if they are not already present) and place the result in the variable "number", using at least 2 integer digits.

<%@if "Enter your name"="My name"%>
...
<%@else%>
...
<%@endif%>
If the variable "Enter your name" has the value "My name" (or has the value of the variable "My name") then import the items represented by the first ... (else import the items represented by the second ... when present).

<%[pe command]...[pe command]%>
Compile the commands and add them to the macro.

Simple usage of import command using the installed "header.txt" file:

This file helps the user to build a C++ header soliciting a program name and author.

It uses PE32 commands to set the date and time of creation and to rename the current file to the program name and .cpp extension.

Commands to execute:

1) e dummy
2) import HEADER.TXT

Explanation of the commands included in the "header.txt" file:

a) Program name is inserted into line 2. Note that a minimum of 20 characters are inserted and the program name is aligned to the left by padding with blanks if necessary.

b) Author name is defaulted to "Your name" and is inserted into line 4. Note that a minimum of 20 characters are inserted and the author name is aligned to the left by padding with blanks if necessary.

c) The current date and time are inserted in the header files without the user being prompted.

d) The file is renamed to the inserted program name.cpp Note that the minimum length of the program name string was changed to 1 from the original 20 characters effectively suppressing any padding with blanks.

Command aliases available :

s = set
n = name
m = macro
e = edit
v = view
q = quit
c = change
l = locate
/ = locate
d = def
k = key
sys = system
h = help
? = help
ve = vbsexec
vr = vbsrun
vt = vbstimeout
vf = vbsrefresh
px = pyexec
pc = pycall
py = pyrun
<nr> = line <nr>
o = open
i = import
w = workspace

Personal Editor 32: PROFILE FILE INSTRUCTIONS

All the following commands may be used on the command line (if included in square brackets) or used in the profile file after a key or syn definition (def = [...] or syn = [...]). They have to be included between square brackets.

If some commands present in the profile file are not included in def or syn commands they are executed when the profile file is loaded (only if a file is already opened).

All commands that have a square bracket as part of a parameter can be included in double quotes (e.g. ["l/]/"]).

All empty profile file lines are ignored and all lines beginning with an asterisk (* remark) are treated as remarks.

All commands may be typed in the short form or in the extended form (see the table below).

Some commands may be followed by a character enclosed in single quotes ('') to supply the value that would otherwise be requested. Such commands include fill mark and all bookmark commands.

For example:

[fill mark] '$' or [fill mark '$']
[bookmark set] 'a' or [bookmark set 'a']

Profile file instructions table:

Short form	Extended form	Command description
ag	align	Align the current line to the cursor position
am	align mark	Align all marked lines to the cursor position
at	align tab	Align the current line to the next tab position
ak	align backtab	Align the current line to the previous tab position
ao	align block	Align the mark to the beginning of the block
al	append line	Append a line after current line
ab	append to clipboard	Append the marked text to the clipboard
au	autocomplete	Complete the word with the most similar
bt	backtab	Move cursor to the previous tab position
bw	backtab word	Move cursor to the previous word

Short form	Extended form	Command description
bp	beep	Execute a beep
bl	begin line	Move cursor to the beginning of the line
bm	begin mark	Move cursor to the beginning of the mark
wb	begin word	Move cursor to the beginning of the word
bs	bookmark set	Set a bookmark at the cursor position
bg	bookmark goto	Move cursor to the typed bookmark
br	bookmark remove	Remove one bookmark
bo	bottom	Move cursor to the bottom of the file
ba	bottom all	Move cursor to the bottom on all files
be	bottom edge	Move cursor to the bottom of the window
bk	begin keyword	Move cursor to the begin of the keyword
bu	begin unknown	Move cursor to the begin of the unknown word
bi	begin string	Move cursor to the begin of the string
bn	begin number	Move cursor to the begin of the number
bb	begin remark	Move cursor to the begin of the remark
bq	begin separator	Move cursor to the begin of the separator
bc	block from clipboard	Copy the clipboard text as a block mark
kc	capitalize	Capitalize the case of all marked chars
ci	center in margins	Center the line in the margins
cl	center line	Scroll the line to the window center
ck	clear marks	Clear the current mark and the mark stack
ce	clear message	Clear the message in the status line
cn	colnum	Insert the column number in the file
cg	command toggle	Toggle cursor from command to text
co	confirm change	Ignored

Short form	Extended form	Command description
ch	context help	Open the help file on the current word
cf	copy from command	Copy the command line at the cursor position
cp	copy from clipboard	Copy the clipboard at the cursor position
cm	copy mark	Copy the mark at the cursor position
cb	copy to clipboard	Copy the marked text into the clipboard
cr	create table	Create a table using delimiter and marked lines
ct	copy to command	Copy the current line into command line
cc	cursor command	Move the cursor into the command line
ca	cursor data	Move the cursor into the text window
dt	date	Insert the current date (see set date)
da	delete all	Delete all file lines
dc	delete char	Delete the current char
dj	delete join	Delete the current char and join at end
dl	delete line	Delete the current line
dm	delete mark	Delete the current mark
dw	delete word	Delete the current word
de	demo end	Disable the demo mode
df	demo fast	Enable the demo fast mode
ds	demo slow	Enable the demo slow mode
dp	demo step	Enable the demo step mode
dn	down	Move cursor down one line
wl	down all	Move cursor down one line on all files
du	dup line	Duplicate the current line
ed	edit	Open a new file
el	end line	Move cursor to the end of the line

Short form	Extended form	Command description
em	end mark	Move cursor to the end of the mark
we	end word	Move cursor to the end of the word
eb	erase begin line	Erase from cursor to beginning of line
ee	erase end line	Erase from cursor to end of line
es	escape	Insert the typed key as ascii character
ex	execute	Execute the command line command
ep	expand	Run the current word as a synonym
ek	end keyword	Move cursor to the end of the keyword
eo	end unknown	Move cursor to the end of the unknown word
et	end string	Move cursor to the end of the string
en	end number	Move cursor to the end of the number
er	end remark	Move cursor to the end of the remark
ea	end separator	Move cursor to the end of the separator
fm	fill mark	Fill the mark with the typed character
fb	find blank line	Move cursor to the next empty line
fn	first nonblank	Move cursor to the first non blank character
gw	get word	Mark the current word
in	indent	Move cursor to the left margin
il	insert line	Insert an empty line after the current
im	insert mode	Change cursor to insert mode
it	insert toggle	Toggle cursor from insert to overstrike
ic	invertcase	Invert the case of all marked chars
ph	prev history	Show the previous command line
nh	next history	Show the next command line
jo	join	Join the next line to the current

Short form	Extended form	Command description
ju	justify line	Align the line to the right margin
jp	justify paragraph	Align the paragraph to the right margin
ja	justify all	Align all lines to the right margin
lr	learn	Add a keyword to the current .kwd file
lf	left	Move cursor left one char
le	left edge	Move cursor to the screen left edge
lg	left margin	Move cursor to the left margin
ln	linenum	Insert the line number in the file
lt	link to	Open the web site at cursor position
lc	lowercase	Change all marked chars to lower case
kd	macro define	Start/stop recording a command macro
ku	macro pause	Pause the recording of a macro
kp	macro play	Play the last recorded macro
mo	mail to	Send an e-mail to the address at cursor position
mb	mark block	Define start and end of a rect mark
mc	mark char	Define start and end of a stream mark
ml	mark line	Define start and end of a line mark
mt	mark tab	Align all the marked lines to the next tab
mk	mark backtab	Align all the marked lines to the previous tab
ma	match brace	Search for the next brace ()[]{}
mm	move mark	Move the mark to the cursor position
nk	next keyword	Move cursor to the next keyword
no	next unknown	Move cursor to the next unknown word
ns	next string	Move cursor to the next string
nn	next number	Move cursor to the next number

Short form	Extended form	Command description
nr	next remark	Move cursor to the next remark
np	next separator	Move cursor to the next separator
nw	next window	Move cursor to the next view
nv	next view	Move cursor to the same file next view
nu	null	No operation
ob	overlay block	Overlay the mark at the cursor position
pd	page down	Move cursor one page down
pa	page down all	Move cursor one page down on all files
pg	paragraph margin	Move cursor to the paragraph margin
pu	page up	Move cursor one page up
ua	page up all	Move cursor one page up on all files
pe	pathname	Insert the current path name in the file
po	pop mark	Restore a mark from the mark stack
pm	print mark	Print marked chars on the default printer
ps	push mark	Save the current mark in the mark stack
pk	prev keyword	Move cursor to the previous keyword
pw	prev unknown	Move cursor to the previous unknown word
pt	prev string	Move cursor to the previous string
pn	prev number	Move cursor to the previous number
pr	prev remark	Move cursor to the previous remark
pp	prev separator	Move cursor to the previous separator
rh	repeat change	Repeat the last change command executed
rc	repeat command	Repeat the last command executed
rs	repeat locate	Repeat the last locate command executed
re	redo	Redo the last 255 undo commands

Short form	Extended form	Command description
rw	redraw	Refresh the current file window
rf	reflow	Align marked lines to left/right margins
ra	reformat all	Align all lines to left/right margins
rp	reformat paragraph	Align paragraph to left/right margins
rm	replace mode	Change cursor to replace mode
rt	right	Move cursor right one char
rd	right edge	Move cursor to the screen right edge
rg	right margin	Move cursor to the right margin
ro	rubout	Delete the previous character
rj	rubout join	Delete the previous char and join at begin
sd	scrolldown	Scroll the screen one line down
rl	scrollleft	Scroll the screen one column left
rr	scrollright	Scroll the screen one column right
su	scrollup	Scroll the screen one line up
sl	shift left	Shift mark left one char
sr	shift right	Shift mark right one char
sp	split	Split the line at the current position
ss	split screen	Split the screen in 2 different views
tb	tab	Move cursor to the next tab position
tw	tab word	Move cursor to the next word
tm	time	Insert the current time (see set time)
to	top	Move cursor to the top of the file
ta	top all	Move cursor to the top on all files
te	top edge	Move cursor to the top of the window
tl	trim leading	Remove leading blanks from line or mark

Short form	Extended form	Command description
tt	trim trailing	Remove trailing blanks from line or mark
tc	trim compress	Remove multiple blanks from line or mark
ud	undo	Undo the last 255 commands or keys
um	unmark	Deselect the current mark
up	up	Move cursor up one line
ul	up all	Move cursor up one lineon all files
uc	uppercase	Change all marked chars to upper case
zw	zoom window	Return the screen to a single view

These commands can be followed by the number of times to repeat the same function:

[up <nr>]
[down <nr>]
[left <nr>]
[right <nr>]
[scrollup <nr>]
[scrolldown <nr>]
[scrollleft <nr>]
[scrollright <nr>]

These commands can be followed by an expression to use the returned value in the evaluation:

[linenum <expression>]
[colnum <expression>]

If inside the square brackets there is a number only, this is interpreted as the ASCII code of the character to insert in the current file.

e.g. [65] becomes 'A', [12] becomes page eject.

Personal Editor 32: SYNTAX COLORING SETTINGS

Two profile file commands (set syntaxcoloring and set syntaxcoloringext) let you define which files have the syntax coloring feature enabled, and which colors are to be used to draw the different keywords and operators.

The procedure that analyzes the language uses the commands present in the PE32.PRO file and in the PE32.KWD* files to assign the different colors to the keywords.

The only settings needed in the profile files are the file extensions for each language and the colors for each language's keywords/operators.

These settings may be used to specify up to 20 different syntaxes.

The lists of all keywords to colorize are in the PE32.KWD* files.

All keywords in the file PE32.KWD* after the '$$PE32_END_KEYWORDS$$' word are displayed with the color specified for the system keywords (the last 2 parameters in the set syntaxcoloring command).

Some other settings may be present in the keyword file to match the
syntax rules:

After $$PE32_SEPARATORS= type all separators used by the syntax.

After $$PE32_REMARK_BEGIN= type one or more characters to define the keyword that starts a block remark.

After $$PE32_REMARK_END= type one or more characters to define the keyword that ends a block remark.

After $$PE32_REMARK_LINE= type one or more characters to define the keyword that ends the line and starts a trailing comment.

After $$PE32_REMARK_COLUMN= type the column number where to search for the characters defined in $$PE32_REMARK_LINE command, if you don't specify a column number these characters are searched in all the line.

After $$PE32_REMARK_AFTER= type the column number where the remarks are starting or leave it to zero.

After $$PE32_SEQUENCE_BEFORE= type the column number where ends the sequence lines numbering (starting from the first column). This field is colored as the number field defined in the syntaxcoloring command.

With $$PE32_IGNORE_CASE$$ you can cause the case of the keywords in the file to be ignored.

After $$PE32_STRINGS= type one or more characters to define the string delimiters.

If you leave any of the above settings out of a keyword file, then the default settings match the C++ language rules and are as follows:

```
$$PE32_SEPARATORS=(){}[]<>+-*/=%!|^&~,.;:?
```

```
$$PE32_REMARK_BEGIN=/*
$$PE32_REMARK_END=*/
$$PE32_REMARK_LINE=//
$$PE32_STRINGS="'
```

The default keyword files installed with Personal Editor 32 are:

- PE32.KWD : C/C++, MFC and Windows API keywords file
- PE32.KWD1 : HTML keywords file
- PE32.KWD2 : National language dictionary (see below)
- PE32.KWD3 : Dos batch files keywords file
- PE32.KWD4 : PE32 profile keywords file
- PE32.KWD5 : Visual Basic script keywords file
- PE32.KWD6 : Assembler keywords file
- PE32.KWD7 : Visual Basic keywords file
- PE32.KWD8 : Java script keywords file
- PE32.KWD9 : Cobol script keywords file

You may change your profile file in order to modify, add or remove the syntax coloring settings to choose the best editor configuration.

All the national languages dictionaries available are present in the web site and can be downloaded and copied in the installation directory changing the name in PE32.KWD2 in order to enable the national words spell checker.

Personal Editor 32: VISUAL BASIC SCRIPT USE

What Is VBScript?

Microsoft Visual Basic Scripting Edition, the newest member of the Visual Basic family of programming languages, brings active scripting to a wide variety of environments, including Web client scripting in Microsoft Internet Explorer and Web server scripting in Microsoft Internet Information Service.

Easy to Use and Learn

If you already know Visual Basic or Visual Basic for Applications, VBScript will be very familiar. But even if you don't know Visual Basic, once you learn VBScript, then programming with the whole family of Visual Basic languages will then be easy.

VBScript Features:

Array handling
Assignments
Comments
Control Flow Instructions
Conversions
Declarations
Error Handling
Expressions
File System Access
Formatting Strings
Input/Output

Miscellaneous Functions
Regular Expressions
Objects
Operators
Options
Procedures
Variants

VBScript use in Personal Editor 32

There are two ways to use VBScript in Personal Editor 32: execute one or more VBScript commands and procedures or execute a complete VBScript program.

From the editor command line use the first method by using the VbsExec command followed by one or more commands, or use the second method by using the VbsRun command, followed by the program file name.

All VBScript commands and VBScript programs must be legal syntactically in order to work properly. When this is not true, a message box with the problem and it's line and column positions will help you fix the problem.

These sample commands can be typed on the command line:

```
vbsexec msgbox "Message Box Test"

vbsexec var=10:string="abc":msgbox "Var:"&var&" String:"&string

vbsexec NewProcedure

vbsrun c:\program.vbs
```

This sample procedure can be placed in the PE32.VBS file.
Execute it by typing "VbsExec RemFunct" on the command line.

```
'
'From current position until matching C block depth
'  comment out (add // to start of line) all source lines.
'
'Note:  Will comment all lead-in lines before first block mark ({).
'
function RemFunct

Dim regEx, Matches
Set regEx = New RegExp' Use regular expression
regEx.IgnoreCase = True' Set ignore case
regEx.Global = True' Set global search

brace=0
exitloop=0

'  Loop over block (or until EOF)
do' Loop until end

str=pe32.line' Get pe32 current line

'Increment count for blocks opened
regEx.Pattern = "{" ' Set pattern
Set Matches = regEx.Execute(str)' Execute search
brace=brace+Matches.count' Count braces

' Decrement count for blocks closed
regEx.Pattern = "}" ' Set pattern
Set Matches = regEx.Execute(str)' Execute search
brace=brace-Matches.count' Count braces

' Comment the line
str="// "+str' Add a C remark
pe32.line=str' Set current pe32 line
newrow=pe32.row+1' Increment row

' Exit if EOF
if newrow>pe32.lines then' If eof exit
exitloop=1
end if

' Exit at matching depth
```

```
if brace<=0 and Matches.count>0 then ' If end function exit
exitloop=1
end if

pe32.row=newrow ' Set new row

loop until exitloop=1 ' Repeat loop

end function
```

As can seen within the VBScript program you can read and write all Personal Editor 32 properties and call all of its functions to retrieve lines, characters, marked lines, filenames, etc.

The following Personal Editor 32 functions are available from VBScript:

pe32.Command "<string var>"

Execute a Personal Editor 32 command line instruction or a profile file command.

This may not include macro commands such as [key <key> <nr>], [demo ...], [* Comment], or [change] nor vbsrun or vbsexec commands.
However the non-macro commands change/.../ and * Comment work.

<string var> = pe32.GetLine(<numeric var>)

Return in a string variable the line with the number in the numeric variable.

pe32.SetLine "<string var>", <numeric var>
Set the line with the number in the numeric variable with the value in the string variable.

<string var> = pe32.GetMarkedLine(<numeric var>)
Return in a string variable the marked line with the number in the numeric variable.

pe32.SetMarkedLine "<string var>", <numeric var>
Set the marked line with the number in the numeric variable with the value in the string variable.

<string var> = pe32.Get("<set command>")
Return the current setting for any of the PE32 set commands.
Write as parameter the command without "set" (e.g.: "tabs").

<string var> = pe32.GetString("<string title>", "<string default>")

Ask for a string in a message box with a title and a default string and return the typed string in the variable.

<string var> = pe32.GetEnv("<string variable>")
Return in the string variable the content of the Dos environment variable.

This is the list of Personal Editor 32 properties available from VBScript:

<numeric var> = pe32.Row
Return the current row number.

pe32.Row = <numeric var>
Set the current row number.

<numeric var> = pe32.Col
Return the current column number.

pe32.Col = <numeric var>
Set the current column number.

<numeric var> = pe32.Lines
Return the total number of lines in the file.

<numeric var> = pe32.MarkedLines
Return the number of marked lines.
The mark can be in any opened file.

<numeric var> = pe32.InsertMode
Return the current insert mode (0 or 1).

pe32.InsertMode = <numeric var>
Set the current insert mode (0 or 1).

<numeric var> = pe32.CommandMode
Return the current command mode (0 or 1).

pe32.CommandMode = <numeric var>
Set the current command mode (0 or 1).

<string var> = pe32.Filename
Return the file name

pe32.Filename = "<string var>"
Set the file name.

<string var> = pe32.Line
Return the current line.

pe32.Line = "<string var>"
Set the current line.

<string var> = pe32.Word
Return the current word.

pe32.Word = "<string var>"
Set the current word.

pe32.Insert = "<string var>"
Insert the text at the current position.

<numeric var> = pe32.Key
Return the typed key.

<numeric var> = pe32.Shift
Return the keyboard shift state using a bit for each of the following:

right alt pressed	0x0001 hexadecimal
left alt pressed	0x0002 hexadecimal
right ctrl pressed	0x0004 hexadecimal
left ctrl pressed	0x0008 hexadecimal
shift pressed	0x0010 hexadecimal
num lock on	0x0020 hexadecimal
scroll lock on	0x0040 hexadecimal
caps lock on	0x0080 hexadecimal
enhanced key	0x0100 hexadecimal

<numeric var> = pe32.Found
Return the last search or replace result.

pe32.Found = <numeric var>
Set the search result. If set to 0 the current macro is aborted.

<numeric var> = pe32.Modified
Return the current file modification state.

pe32.Modified = <numeric var>
Set the current file modification state.

<numeric var> = pe32.ReadOnly
Return the current file read only state.

pe32.ReadOnly = <numeric var>
Set the current file read only state.

<numeric var> = pe32.Encrypted
Return the current file encryption state.

pe32.Encrypted = <numeric var>
Set the current file encryption state.

<numeric var> = pe32.MarkType
Return the current mark type as:
0 = no mark
if mark is in the current file:
1 = line mark
2 = block mark
3 = stream mark
4 = line mark in the command line
5 = block mark in the command line
6 = stream mark in the command line
if mark is not in the current file:
9 = line mark
10 = block mark
11 = stream mark
12 = line mark in the command line
13 = block mark in the command line

14 = stream mark in the command line

pe32.Message = "<string var>"
Write the string in the status line.

<numeric var> = pe32.CommandCol
Return the command line column number.

pe32.CommandCol = <numeric var>
Set the command line column number.

VBScript features are available on all systems where Internet Explorer is installed. However, you have to install and register two additional files to use VBScript from Personal Editor 32.

See the "INSTALLATION AND USE" paragraph for more details.

For more information about VBScript features and syntax see Microsoft Visual Basic documentation at:

http://msdn.microsoft.com/en-us/library/ms950396.aspx
http://msdn.microsoft.com/en-us/library/d1et7k7c(VS.85).aspx
http://msdn.microsoft.com/en-us/library/t0aew7h6(VS.85).aspx

To use VBScript with Personal Editor 32 you need an OCX control called MSSCRIPT.OCX which is included in the installation package and in the operating systems from Windows 2000 and above.

Personal Editor 32: PYTHON LANGUAGE USE

What Is Python?

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

The Python version integrated in PE32 is the 2.7, please refer to the www.python.org site to download and install the language.

Python use in Personal Editor 32

There are three ways to use Python in Personal Editor 32: execute one commands, call a procedure or execute a complete Python program.

From the editor command line use the first method by using the PyExec command followed by one command, use the second method by using the PyCall command, followed by the procedure name and its parameters, or use the PyRun command, followed by the program file name.

All Python commands and Python programs must be legal syntactically in order to work properly. When this is not true, a message box with the problem and it's line and column positions will help you fix the problem. Please remember that all the Python commands are case sensitive.

These sample commands can be typed on the command line:

```
pyexec a=100

pycall Procedure(100)

pyrun c:\program.py
```

Within the Python programs you can read and write all the Personal Editor 32 properties and call all of its functions to retrieve lines, characters, marked lines, filenames, etc.

The following Personal Editor 32 functions are available from Python:

. pe32.Command "<string var>" : Execute a Personal Editor 32 command line instruction or a profile file command.

This may not include macro commands such as [key <key> <nr>], [demo ...], [* Comment], or [change] nor pyrun or pyexec commands.

However the non-macro commands change/.../
and * Comment work.

. <string var> = pe32.GetLine(<numeric var>) : Return in a string variable
the line with the number in the numeric
variable.

. pe32.SetLine "<string var>", <numeric var> : Set the line with the number
in the numeric variable with the value in the
string variable.

. <string var> = pe32.GetMarkedLine(<numeric var>) : Return in a string
variable the marked line with the number in
the numeric variable.

. pe32.SetMarkedLine "<string var>", <numeric var> : Set the marked line with
the number in the numeric variable with the
value in the string variable.

. <string var> = pe32.Get("<set command>") : Return the current setting for
any of the PE32 set commands.
Write as parameter the command
without "set" (e.g.: "tabs").

. <string var> = pe32.GetString("<string title>", "<string default>") :
Ask for a string in a message box with a title
and a default string and return the typed
string in the variable.

. <string var> = pe32.GetEnv("<string variable>") : Return in the string
variable the content of the Dos environment
variable.

This is the list of Personal Editor 32 properties available from Python:

. <numeric var> = pe32.Row : Return the current row number.

. pe32.Row = <numeric var> : Set the current row number.

. <numeric var> = pe32.Col : Return the current column number.

. pe32.Col = <numeric var> : Set the current column number.

. <numeric var> = pe32.Lines : Return the total number of lines in the file.

. <numeric var> = pe32.MarkedLines : Return the number of marked lines.
The mark can be in any opened file.

. <numeric var> = pe32.InsertMode : Return the current insert mode (0 or 1).

. pe32.InsertMode = <numeric var> : Set the current insert mode (0 or 1).

. <numeric var> = pe32.CommandMode : Return the current command mode (0 or
1).

. pe32.CommandMode = <numeric var> : Set the current command mode (0 or 1).

. <string var> = pe32.Filename : Return the file name

. pe32.Filename = "<string var>" : Set the file name.

. <string var> = pe32.Line : Return the current line.

. pe32.Line = "<string var>" : Set the current line.

. <string var> = pe32.Word : Return the current word.

. pe32.Word = "<string var>" : Set the current word.

. pe32.Insert = "<string var>" : Insert the text at the current position.

. <numeric var> = pe32.Key : Return the typed key.

. <numeric var> = pe32.Shift : Return the keyboard shift state using a
bit for each of the following:

right alt pressed	0x0001 hexadecimal
left alt pressed	0x0002 hexadecimal
right ctrl pressed	0x0004 hexadecimal
left ctrl pressed	0x0008 hexadecimal
shift pressed	0x0010 hexadecimal
num lock on	0x0020 hexadecimal
scroll lock on	0x0040 hexadecimal
caps lock on	0x0080 hexadecimal
enhanced key	0x0100 hexadecimal

. <numeric var> = pe32.Found : Return the last search or replace result.

. pe32.Found = <numeric var> : Set the search result. If set to 0 the
current macro is aborted.

. <numeric var> = pe32.Modified : Return the current file modification state.

. pe32.Modified = <numeric var> : Set the current file modification state.

. <numeric var> = pe32.ReadOnly : Return the current file read only state.

. pe32.ReadOnly = <numeric var> : Set the current file read only state.

. <numeric var> = pe32.Encrypted : Return the current file encryption state.

. pe32.Encrypted = <numeric var> : Set the current file encryption state.

. <numeric var> = pe32.MarkType : Return the current mark type as:
0 = no mark
if mark is in the current file:
1 = line mark
2 = block mark
3 = stream mark
4 = line mark in the command line
5 = block mark in the command line

6 = stream mark in the command line
if mark is not in the current file:
9 = line mark
10 = block mark
11 = stream mark
12 = line mark in the command line
13 = block mark in the command line
14 = stream mark in the command line

. pe32.Message = "<string var>" : Write the string in the status line.

. <numeric var> = pe32.CommandCol : Return the command line column number.

. pe32.CommandCol = <numeric var> : Set the command line column number.

Personal Editor 32: Keys Name Table

List of PE32 available keys name you can use in the def command:

Key Name	Normal	Shift (s)	Ctrl (c)	Alt (a)	Shift+Ctrl (u)	Ctrl + Alt (x)
"up"	yes	yes	yes	no	yes	yes
"down"	yes	yes	yes	no	yes	yes
"left"	yes	yes	yes	no	yes	yes
"right"	yes	yes	yes	no	yes	yes
"pgup"	yes	yes	yes	no	yes	yes
"pgdn"	yes	yes	yes	no	yes	yes
"home"	yes	yes	yes	no	yes	yes
"end"	yes	yes	yes	no	yes	yes
"ins"	yes	yes	yes	no	yes	yes
"insert" (**)	yes	yes	yes	no	yes	yes
"del"	yes	yes	yes	yes	yes	yes
"enter"	yes	no	yes	no	yes	yes
"backspace"	yes	no	yes	no	yes	yes
"space"	yes	no	yes	no	yes	yes
"tab"	yes	yes	yes	no	yes	yes
"a"	yes	yes	yes	yes	yes	yes
"b"	yes	yes	yes	yes	yes	yes
"c"	yes	yes	yes	yes	yes	yes
"d"	yes	yes	yes	yes	yes	yes
"e"	yes	yes	yes	yes	yes	yes
"f"	yes	yes	yes	yes	yes	yes
"g"	yes	yes	yes	yes	yes	yes
"h"	yes	yes	yes	yes	yes	yes
"i"	yes	yes	yes	yes	yes	yes
"j"	yes	yes	yes	yes	yes	yes
"k"	yes	yes	yes	yes	yes	yes
"l"	yes	yes	yes	yes	yes	yes
"m"	yes	yes	yes	yes	yes	yes
"n"	yes	yes	yes	yes	yes	yes
"o"	yes	yes	yes	yes	yes	yes
"p"	yes	yes	yes	yes	yes	yes
"q"	yes	yes	yes	yes	yes	yes
"r"	yes	yes	yes	yes	yes	yes
"s"	yes	yes	yes	yes	yes	yes

Key Name	Normal	Shift (s)	Ctrl (c)	Alt (a)	Shift+Ctrl (u)	Ctrl + Alt (x)
"t"	yes	yes	yes	yes	yes	yes
"u"	yes	yes	yes	yes	yes	yes
"v"	yes	yes	yes	yes	yes	yes
"w"	yes	yes	yes	yes	yes	yes
"x"	yes	yes	yes	yes	yes	yes
"y"	yes	yes	yes	yes	yes	yes
"z"	yes	yes	yes	yes	yes	yes
"f1"	yes	yes	yes	yes	yes	yes
"f2"	yes	yes	yes	yes	yes	yes
"f3"	yes	yes	yes	yes	yes	yes
"f4"	yes	yes	yes	yes	yes	yes
"f5"	yes	yes	yes	yes	yes	yes
"f6"	yes	yes	yes	yes	yes	yes
"f7"	yes	yes	yes	yes	yes	yes
"f8"	yes	yes	yes	yes	yes	yes
"f9"	yes	yes	yes	yes	yes	yes
"f10"	yes	yes	yes	yes	yes	yes
"f11"	yes	yes	yes	yes	yes	yes
"f12"	yes	yes	yes	yes	yes	yes
"0"	yes	yes	yes	yes	yes	yes
"1"	yes	yes	yes	yes	yes	yes
"2"	yes	yes	yes	yes	yes	yes
"3"	yes	yes	yes	yes	yes	yes
"4"	yes	yes	yes	yes	yes	yes
"5"	yes	yes	yes	yes	yes	yes
"6"	yes	yes	yes	yes	yes	yes
"7"	yes	yes	yes	yes	yes	yes
"8"	yes	yes	yes	yes	yes	yes
"9"	yes	yes	yes	yes	yes	yes
"minus"	yes	yes	yes	yes	yes	yes
"equals"	yes	yes	yes	yes	yes	yes
"lbracket"	yes	yes	yes	yes	yes	yes
"rbracket"	yes	yes	yes	yes	yes	yes
"backslash"	yes	yes	yes	yes	yes	yes
"semi"	yes	yes	yes	yes	yes	yes
"semicolon" (*)	yes	yes	yes	yes	yes	yes
"backquote"	yes	yes	yes	yes	yes	yes

Key Name	Normal	Shift (s)	Ctrl (c)	Alt (a)	Shift+Ctrl (u)	Ctrl + Alt (x)
"comma"	yes	yes	yes	yes	yes	yes
"quote"	yes	yes	yes	yes	yes	yes
"padminus"	yes	yes	yes	yes	yes	yes
"padplus"	yes	yes	yes	yes	yes	yes
"slash"	yes	yes	yes	yes	yes	yes
"period"	yes	yes	yes	yes	yes	yes
"pad5"	yes	yes	yes	no	yes	yes
"pad*"	yes	no	yes	no	yes	yes
"esc"	yes	yes	no	no	no	no
"prtsc"	no	no	no	no	no	no
"exclamationp"	yes	no	no	no	no	no
"doublequote"	yes	no	no	no	no	no
"numbers"	yes	no	no	no	no	no
"dollars"	yes	no	no	no	no	no
"percents"	yes	no	no	no	no	no
"ampersand"	yes	no	no	no	no	no
"lpar"	yes	no	no	no	no	no
"rpar"	yes	no	no	no	no	no
"asterisk"	yes	no	no	no	no	no
"plus"	yes	no	no	no	no	no
"colon"	yes	no	no	no	no	no
"lesst"	yes	no	no	no	no	no
"greatert"	yes	no	no	no	no	no
"questionm"	yes	no	no	no	no	no
"ats"	yes	no	no	no	no	no
"carets"	yes	no	no	no	no	no
"underscore"	yes	no	no	no	no	no
"lbrace"	yes	no	no	no	no	no
"pipes"	yes	no	no	no	no	no
"rbrace"	yes	no	no	no	no	no
"tilde"	yes	no	no	no	no	no

(*): "semi" and "semicolon" are alias
(**): "ins" and "insert" are alias

Personal Editor 32: Commands Reference

Command	Abbr.	Description	Menu
[<3 digits number>]	[nnn]	convert the number in ascii character	
about	[ver]	displays the PE32 ABOUT box	HELP.....about
align	[ag]	align line to cursor	
align backtab	[ak]	align line to previous tab	
align block	[ao]	align mark to beginning of block	
align mark	[am]	align marked lines to cursor	
align tab	[at]	align line to next tab	
append <filename>		append marked text to specified file	
append line	[al]	insert line after current line	
append to clipboard	[ab]	append marked text to clipboard	
autocomplete	[au]	autocomplete with most similar word	
backtab	[bt]	cursor to previous tab	
backtab word	[bw]	cursor to previous word	SEARCH...prev word
beep	[bp]	beep	
begin keyword	[bk]	cursor to begin of keyword	
begin line	[bl]	cursor to begin line	SEARCH...begin line
begin mark	[bm]	cursor to begin mark	MARK.....begin mark
begin number	[bn]	cursor to begin of number	
begin remark	[bb]	cursor to begin of remark	
begin separator	[bq]	cursor to begin of separator	
begin string	[bi]	cursor to begin of string	
begin unknown	[bu]	cursor to begin of unknown word	
begin word	[wb]	cursor to begin word	SEARCH...begin word
block from clipboard	[bc]	copy from clipboard as block	
bookmark goto	[bg]	cursor to bookmark	SEARCH...bookmark goto
bookmark remove	[br]	remove one bookmark	SEARCH...bookmark remove
bookmark set	[bs]	set bookmark at cursor	SEARCH...bookmark set
bookmarks		dialog box of bookmarks	SEARCH...bookmarks list
bottom	[bo]	cursor to end of file	WINDOW...end file
bottom all	[ba]	cursor to end of all files	WINDOW...end all files
bottom edge	[be]	cursor to window bottom	WINDOW...bottom edge
calc <expression>[=$row,col]		calculate result of expression	TOOLS....calculate
capitalize	[kc]	change mark to capitals	
cd [<path>]		change directory	FILE.....cd
center in margins	[ci]	align line to center in margins	TEXT.....center in margins
center line	[cl]	scroll line to window center	TEXT.....center line
change/old/new/[*-mnewpaxoichtb#]	[c]	change text according to parameters	SEARCH...change
chars		count chars in mark or file	
clear marks	[ck]	clear the current mark and the mark stack	
clear message	[ce]	delete status line message	

Command	Abbr.	Description	Menu
colnum <expression>	[cn]	insert column number in file	
column <number>		cursor to column specified	SEARCH...goto column
command toggle	[cg]	cursor command line to text toggle	
confirm change	[co]	ignored	
context help	[ch]	open help file on current word	HELP.....context help
copy from clipboard	[cp]	copy from clipboard at cursor position	MARK.....from clipboard
copy from command	[cf]	copy from command line at cursor	TOOLS....copy from command
copy mark	[cm]	copy mark to cursor position	MARK.....copy
copy to clipboard	[cb]	copy mark to clipboard	MARK.....to clipboard
copy to command	[ct]	copy current line to command line	TOOLS....copy to command
create table	[cr]	create table using delimiters and marked lines	
cursor command	[cc]	cursor to command line	
cursor data	[ca]	cursor to text window	
date	[dt]	insert date	
def <key> = [command]...[command]	[d]	define key as sequence of commands	
delete all	[da]	delete all file lines	TEXT.....delete all
delete character	[dc]	delete current char	
delete join	[dj]	delete current char and join at end	
delete line	[dl]	delete current line	TEXT.....delete line
delete mark	[dm]	delete current marked characters	MARK.....delete
delete word	[dw]	delete current word	TEXT.....delete word
demo end	[de]	demo disable	
demo fast	[df]	demo fast	
demo slow	[ds]	demo slow	
demo step	[ds]	demo step	
dir [<pathname>] <filters...>		dialog box of files in directory	FILE.....dir
dos <command> <parameters> <...>		execute dos command or open DOS window	TOOLS....dos prompt
down <nr>	[dn]	cursor down one or nr lines	
down all	[wl]	cursor down one line on all files	
dup line	[du]	insert copy of current line below current line	TEXT.....dup line
edit [<filename>] [mri]	[e]	open a new file with optional parameters	FILE.....open
end keyword	[ek]	cursor to end keyword	
end line	[el]	cursor to end of line	SEARCH...end line
end mark	[em]	cursor to end of mark	MARK.....end mark
end number	[en]	cursor to end number	
end remark	[er]	cursor to end remark	
end separator	[ea]	cursor to end of separator	
end string	[et]	cursor to end string	
end unknown	[eo]	cursor to end unknown	
end word	[we]	cursor to end of word	SEARCH...end word
erase <file>		delete file	FILE.....erase
erase begin line	[eb]	delete cursor to begin line	TEXT.....erase to begin

Command	Abbr.	Description	Menu
			line
erase end line	[ee]	delete from cursor to end of line	TEXT.....erase to end line
escape	[es]	insert typed key as ascii char	
execute	[ex]	execute command line	TOOLS....execute
expand	[ep]	run current word as synonym	
file [<filename>] [<tabs/notabs>]		save file and close window	FILE.....file
fill mark	[fm]	change marked chars to typed character	MARK.....fill
find blank line	[fb]	cursor to next empty line	SEARCH...find blank line
first nonblank	[fn]	cursor to first nonblank	SEARCH...find nonblank
get word	[gw]	mark current word	
help [<set> or <key> or <synonym>]	[h]	display key definition or commands syntax	HELP.....help
history		dialog box of command history	COMMAND..command history
html <filename>		convert to new html file	
import <filename>	[i]	insert data from specified file into current	
indent	[in]	cursor to left margin	
insert line	[il]	insert line after current line	TEXT.....insert line
insert mode	[im]	set insert on	
insert toggle	[it]	set insert <on/off>	
invertcase	[ic]	marked characters invert case	
join	[jo]	join the next line to current one	TEXT.....join line
justify all	[ja]	align all lines to right margin	
justify line	[ju]	align line to right margin	TEXT.....justify line
justify paragraph	[jp]	align paragraph to right margin	TEXT.....justify paragraph
key <key> [<nr>]	[k]	execute assigned commands <nr> times	COMMAND..key assignment
learn	[lr]	add a keyword to .kwd file	
left <nr>	[lf]	cursor left one or nr chars	
left edge	[le]	cursor to left screen edge	
left margin	[lg]	cursor to left margin	
line <row> <column>		cursor to line and column specified	SEARCH...goto line
linenum <expression>	[ln]	insert linenumber in file at cursor	
lines		count lines in mark or file	
link to	[lt]	open the web site at cursor position	
load <filename>		macro load	COMMAND..macro load
locate/string/[*-mnewsaxofichtbr#lg]	[l]	show occurrences of search string	SEARCH...locate
lowercase	[lc]	change marked character to lowercase	MARK.....lower case
macro <profile pathname>	[m]	activate specified profile	COMMAND..load profile
macro define	[kd]	macro recording start/stop	COMMAND..macro record

Command	Abbr.	Description	Menu
macro pause	[ku]	macro recording pause	COMMAND..macro pause
macro play	[kp]	macro play last recorded	COMMAND..macro play
mail <address> "object" [attachment]		sends email to specified address	TOOLS....send mail
mail to	[mo]	send an e-mail to the address at cursor position	
mark backtab	[mk]	align marked line to previous tab	
mark block	[mb]	mark corner of rectangular block	MARK.....block
mark char	[mc]	mark end of stream	MARK.....char
mark line	[ml]	mark line	MARK.....line
mark tab	[mt]	align marked line to next tab	
match brace	[ma]	cursor to search for next brace	SEARCH...match brace
menu		open menu to execute commands	
move mark	[mm]	mark move to cursor position	MARK.....move
mru		dialog box of recently used files	FILE.....recently used
name <file>	[n]	rename current file	FILE.....name
next history	[nh]	show next command line	COMMAND..next command
next keyword	[nk]	cursor to next keyword	
next number	[nn]	cursor to next number	
next remark	[nr]	cursor to next remark	
next separator	[np]	cursor to next separator	
next string	[ns]	cursor to next string	
next unknown	[no]	cursor to next unknown	
next view	[nv]	cursor to next view	WINDOW...next view
next window	[nw]	cursor to next window	WINDOW...next window
null	[nu]	no operation	
open	[o]	dialog box of open files	
overlay block	[ob]	mark overlay at cursor position	MARK.....overlay
page down	[pd]	cursor to next page down	
page down all	[pa]	cursor to next page down all files	
page up	[pu]	cursor to next page up	
page up all	[ua]	cursor to next page up all files	
paragraph margin	[pg]	cursor to paragraph margin	
pathname	[pe]	insert pathname at cursor	
pe32 [/s] <file> [commands] ...		execute another copy of PE32	
pop mark	[po]	mark restored from stack	TOOLS....pop mark
prev history	[ph]	show previous command line	COMMAND..prev command
prev keyword	[ps]	cursor to previous keyword	
prev number	[pn]	cursor to previous number	
prev remark	[pr]	cursor to previous remark	
prev separator	[pp]	cursor to previous separator	
prev string	[pt]	cursor to previous string	
prev unknown	[uw]	cursor to previous unknown	
print		print whole file on default printer	PRINT....file
print mark	[pm]	print marked characters	PRINT....mark

Command	Abbr.	Description	Menu
printdoc [m]		print whole file or marked lines	PRINT....print doc
push mark	[ps]	mark to save in stack	TOOLS....push mark
pyexec <Python commands>	[px]	execute Python commands	TOOLS....python execute
pycall <Python procedures>	[pc]	execute Python procedures	TOOLS....python call
pyrun <filename>	[py]	run python file	TOOLS....python run
quit [all|*] [y]	[q]	quit and close window	FILE.....quit
redo	[re]	redo the last 255 undo commands	COMMAND..redo
redraw	[rw]	refresh the current window	
reflow	[rf]	align marked lines to l/r margins	
reformat all	[ra]	align all lines to l/r margins	TEXT.....reformat all
reformat paragraph	[rp]	align paragraph to l/r margins	TEXT.....reformat paragraph
rename <oldfile> <newfile>		rename the file <oldfile> as <newfile>.	
repeat change	[rh]	repeat last change command	
repeat command	[rc]	repeat last command	
repeat locate	[rs]	repeat last locate command	
replace mode	[rm]	set insert off	
right <nr>	[rt]	cursor right one or nr chars	
right edge	[rd]	cursor to right edge	
right margin	[rg]	cursor to right margin	
rtf <filename>		convert to new rtf file	
rubout	[ro]	delete previous char	
rubout join	[rj]	delete previous char and join at begin	
run <filename>		macro run	COMMAND..macro run
save [<filename>] [<tabs/notabs>]		save file without closing window	FILE.....save
scrolldown <nr>	[sd]	scroll down one or nr lines	
scrollleft <nr>	[rl]	scroll one or nr columns left	
scrollright <nr>	[rr]	scroll one or nr columns right	
scrollup <nr>	[su]	scroll one or nr lines up	
set abbrev <on/off>		set the commands abbreviated form	
set autocompletion <control keys>		set key with space to enable	HELP.....autocomplete
set autosave <value>		set seconds to autosave or 0 to disable	
set backup <extensions...>		set <filename>.BackupPE at save or file	WINDOW...set backup
set bkspinfreespace <on/off>		set backspace in free space	OPTIONS..set bkinfreespace
set blankcompress <on/off>		set blank compression during save	
set borders <characters>		set characters used to draw menu and dialogs	HELP.....set borders
set carriagereturn <on/off>		set cr before lf	
set changeswitch [*-mnewpaxoichtb#]		set change command default switches	
set charset <oem/ansi>		set active codepage	
set codepage <number>		set active codepage	HELP.....set codepage

Command	Abbr.	Description	Menu
set colors <colors...>		set back/fore/dialog colors	WINDOW...set colors
set commandline <on/off> <on/off>		set visibility of command line	OPTIONS..set commandline
set confirm <on/off>		set confirm message during save or file	OPTIONS..set confirm
set currentline <on/off/toggle>		set highlighting of current line	
set currentpos <on/off/toggle>		set highlighting of current position	
set cursor <size> <size>		set cursor size for insert and replace	
set date <string>		set format of date string	OPTIONS..set date
set delimiters <characters...>		set characters to delimit a word	WINDOW...set delimiters
set display <rows> <columns> or max		set display rows and columns	WINDOW...set display
set encryption <on/off>		set encryption during save or file	OPTIONS..set encryption
set eof <on/off		set ctrl-z at end of file	OPTIONS..set end of file
set expandon <control keys>		set key to enable syntax auto expansion	
set helpfile <pathname>		set pathname for context help	HELP.....set helpfile
set historyon <control keys>		set key to show dialog of previous commands	OPTIONS..set historyon
set insert <on/off>		set insert key condition upon load	OPTIONS..set insert
set linefeed <on/off>		set lf after cr	OPTIONS..set linefeed
set linesback <value>		set lines back to search in autocomplete	
set linesforward <value>		set lines forward to search in autocomplete	
set linetrim <off><lead><trail><compr>		set removes blanks on save	OPTIONS..set linetrim
set locateswitch [*-mnewsaxofichtbr#lg]		set locate command default switches	
set margins <left><right><par>< col>		set margins	WINDOW...set margins
set markon <shift/ctrl/alt/off>		set key to mark with arrows	
set oncommand <on/off		set condition of command line upon file open	OPTIONS..set oncommand
set onmenu <on/off>		set opening of command menu upon loading	
set openwith <extensions...>		set extensions to use PE32 by default	
set pecompatibility <on/off>		set the pe compatibility mode	
set popupmenuon <control keys>		set mouse key to open popup menu	
set printer <printername>		set default printer port	PRINT....set printer
set pyload <pathname>		set script file after pe32.py load	
set restorecursor <on/off>		set cursor to last position on loading	OPTIONS..set

Command	Abbr.	Description	Menu
			restorecursor
set showbookmarks <on/off>		set visibility of bookmarks	
set statusline <off/on>		set visibility of status line	OPTIONS..set statusline
set syntaxcoloring<nr> <colors>		set syntax coloring colors	WINDOW...set syntaxcoloring
set syntaxcoloringext<nr> <extensions>		set syntax coloring file extensions	WINDOW...set syntaxcoloringe
set syntaxcoloringpro<nr> <file>		set syntax coloring profile file	
set syntaxcoloringset<nr> <commands>		set syntax coloring set commands	
set tabbed <on/off> <columns>		set tabbed windows for multiple files	
set tabexpand <col>		set expand on load / compress on save	OPTIONS..set tabexpand
set tabinsert <on/off>		set tab indent when insert mode active	
set tabs <col> <step> or <col>..<col>		set tab stops	OPTIONS..set tabs
set time <string>		set format of time string	OPTIONS..set time
set uppercase <on/off>		set uppercase for chars a to z	WINDOW...set uppercase
set vbsload <pathname>		set script file after pe32.vbs load	
set vbsrefresh <on/off>		set screen refresh for VBS	
set vbstimeout <milliseconds>		set timeout for VBS commands or procedures	
set wheelscrollline <lines>		set lines to scroll with mouse wheel	
set wordwrap <on/off>		set word wrap on or off	WINDOW...set wordwrap
shift left	[sl]	shift mark one column left	MARK.....shift left
shift right	[sr]	shift mark one column right	MARK.....shift right
sort [column] [column] [dinrck]		sort lines using optional parameters	TOOLS....sort lines
split	[sp]	insert crlf to split line at cursor	TEXT.....split line
split screen	[ss]	split screen into 2 views	WINDOW...split screen
syn <name> = [command]...[command]		define command names as a synonyms	
system <command> <parameters> <...>	[sys]	execute shell command in same window	
tab	[tb]	cursor to next tab	
tab word	[tw]	cursor to next word	SEARCH...next word
time	[tm]	insert time at cursor	
top	[to]	cursor to top of file	WINDOW...begin file
top all	[ta]	cursor to top in all files	WINDOW...begin all files
top edge	[te]	cursor to top edge	WINDOW...top edge
trim compress	[tc]	delete multiple blanks from line or mark	TEXT.....trim compress
trim leading	[tl]	delete leading blanks from line or mark	TEXT.....trim leading
trim trailing	[tt]	delete trailing blanks from line or mark	TEXT.....trim trailing

Command	Abbr.	Description	Menu
undo	[ud]	undo the last 255 commands or keys	COMMAND..undo
unmark	[um]	clear the current mark	MARK.....unmark
unzip <filename>		extract to ascii file	
up <nr>	[up]	cursor up one or nr lines	
up all	[ul]	cursor up one line all files	
uppercase	[uc]	change marked chars to uppercase	MARK.....upper case
vbsexec <VBS commands>	[ve]	execute vbs commands	TOOLS....vbs execute
vbsrun <filename>	[vr]	run vbs file	TOOLS....vbs run
view [<filename>]	[v]	open file in read-only mode	FILE.....view
words		count words in mark or file	
workspace <filename>	[w]	create workspace file with current settings	FILE.....workspace
write <filename>		macro save to file	COMMAND..macro write
zip <zipfilename>		compress current file and keep open	
zoom window	[zw]	return screen to single view	WINDOW...zoom