Help file produced by

HELLLP!

www.guysoftware.com/helllp.html
(If this topic is presented in file testing the
author should use the HELLLP! **I**
button to define a contents topic)

noyesTRUEnono&AboutC&lose&PrintyesyesyesWinEdit HelpWinEdityes20/02/00

Select an item for display

Help file produced by **HELLLP!** v2.7 , a product of Guy Software, on 02/20/00 for WILSON WINDOWWARE, INC..
The above table of contents will be automatically completed and will also provide an excellent cross-reference for context strings and topic titles. You may leave it as your main table of contents for your help file, or you may create your own and cause it to be displayed instead by using the I button on the toolbar. This page will not be displayed as a topic. It is given a context string of _._ , but this is not presented for jump selection.
HINT: If you do not wish some of your topics to appear in the table of contents as displayed to your users (you may want them ONLY as PopUps), move the lines with their titles and contexts to below this point. If you do this remember to move the whole line, not part. As an alternative, you may wish to set up your own table of contents, see Help under The Structure of a Help File.
Do not delete any codes in the area above the Table of Contents title, they are used internally by HELLLP!

How To ...

- Use command line options
- Use menu scripts
- Write WIL scripts

Wwhatis Wi

WIL Function Reference

Commands

- File menu
- Edit menu
- View menu
- Search menu
- Project menu
- Macro menu
- Window menu
- Help menu

Reference

WIL commands
- WinEdit Command Reference

Page Setup
Print Setup
Print
Print Progress Dialog
Print Preview
Print Preview Toolbar

Debugging WIL Scripts
Toolbars dialog box

Options dialog box
- Options - Editor tab
- Options - File types tab
  - Edit File Types

Syntax coloring
Font Selection
Options - Keyboard tab

Using Regular Expressions

mserror.wbt

Command Line Options

Contacting WinEdit Software Co.
How to get technical support
Registering your software
Ordering Information
ORDER FORM

What is WIL
TUTORIAL
Menu Files
WinEdit Menus
Toolbars
Debug Tool
Color Files
WIL Functions
Project configuration
Recording Macros

wRedo
wUndo()
wPrintDirect()
wProperties()
wViewOptions()
wClearSel()
wSelUp()
wSelDown()
wSelLeft()
wSelRight()
wSelEnd()
wSelHome()
wSelPgUp()
wSelPgDn()
wSelWordLeft()
wSelWordRight()
wSelTop()
wSelBottom()
wStartSel
wEndSel
wInvertCase()
wUpperCase()
wLowerCase()
wCutAppend()
wCutMarked()
wCopyMarked()
wCopyAppend()
wFileSave()
wSetBookmark()
Context Menu
Special Help
WinEdit Key Mappings

WinEdit Menus

Select a menu for further information on menu options

WinEdit Help File

File menu commands

The File menu offers the following commands:

New	Creates a new document.
Open	Opens an existing document.
Previous Files	Displays a list of previously opened documents.
Close	Closes an opened document.
Close All	Closes all opened documents.
Merge	Inserts the contents of a new document into the current document.
Save	Saves an opened document using the same file name.
Save As	Saves an opened document to a specified file name.
Save All	Saves all opened documents.
Revert	Rereads the current document from disk, returning to its original state.
Page Setup	Displays printing options.
Print Setup	Selects a printer and printer connection.
Print	Prints a document.
Print Preview	Displays the document on the screen as it would appear printed.
Send...	Sends the active document through electronic mail.
Properties	Displays information about the current document.
Exit	Exits WinEdit.

Edit menu commands

The Edit menu offers the following commands:

Undo	Reverse previous editing operation.
Redo	Reverse previous undo operation.
Cut	Deletes data from the document and moves it to the clipboard.
Copy	Copies data from the document to the clipboard.
Paste	Pastes data from the clipboard into the document.
Delete	Deletes data from the document.
Copy Other	Copies specific data from the document to the clipboard.
Cut Other	Cuts specific data from the document to the clipboard.
Change Case	Changes the case of selected data from the document.
Select All	Selects all the data in the document.
Column Block	When checked, changes text selection mode to allow column selection.

View menu commands

The View menu offers the following commands:

Toolbars	Shows, hides, or customizes the toolbars.
Status Bar	Shows or hides the status bar.
Directory Tree	Shows or hides the directory tree window.
	In the **Show files of type** edit box, the file type can be changed by adding a new file extension or comma delimited file extension list. To make the changes take effect select the **Refresh** button.
Output	Shows or hides the output window.
Watch	Shows or hides the watch window.
Workbook mode	Toggles Workbook mode on or off. In Workbook mode, each MDI document window has a tab showing the document name. Click on the tab to switch to that window.
Options	Editor, keyboard, and file specific settings are maintained in this dialog box.

Search menu commands

See Also:
Regular Expressions

The Search menu offers the following commands:

Find	Searches the current document for the specified text.
Find next	Repeats the last find operation, using the same options.
Replace	Searches the current document for the specified text, and replaces the found text with specified text.
Find In Files	Searches one or more files for the specified text.
Next Error	Opens the source file with the error or warning and moves the caret to the beginning of the line with the next error or warning. See **Options / File Type** for more information on customizing WinEdit's error parsing.
Previous Error	Moves back to the previous error or warning that was displayed with the Next Error command.
Go To Line	Moves the caret to the specified line number.
Match Brace	If the caret is placed on a brace character (‘‘().{}, or [] ‘‘ the caret is moved to the matching brace character.
Toggle bookmark	Places a bookmark on the current line if one does not already exist, or removes it if it does..
Next bookmark	Moves the caret to the next bookmark.
Previous bookmark	Moves the caret to the previous bookmark.
Clear all bookmarks	Removes all defined bookmarks from the current document.

Project menu commands

The Project menu offers the following commands:

Compile	Executes the Compile command defined in the **Options / File Type** dialog box.
Customize Tools...	Allows a program or WIL script to be added to the Project menu.
Record macro	Starts or stops recording of a WinEdit macro.
Playback recording	Executes the recorded macro.
Save recorded macro	Saves recorded macro as a WIL script file.
Debug macro	Allows interactive debugging of a WIL script file.
New...	Allows the current workspace to be named and saved.
Save as...	Allows the current named workspace to be saved with a new name..
Close	Closes the current workspace, including all document windows and defined tools.
Delete	Allows a named workspace to be deleted.

Recording Macros

You can create a macro by recording it or writing it. The easiest way to create a macro is to record it.

To record a macro

On the Project menu, click Record Macro.
Perform the actions you want to record.
To stop recording, click the Stop button.

To play back the recorded macro, on the Project menu, click Playback Recording.

To save a recorded macro

On the **Project menu,** click **Save Recorded Macro**. You will be prompted for a name for the new macro and WinEdit will, by default, propose to save the macro in the WinEdit\ Macros subdirectory.

After saving the macro, you will be asked if you wish to add the macro to the Project menu.

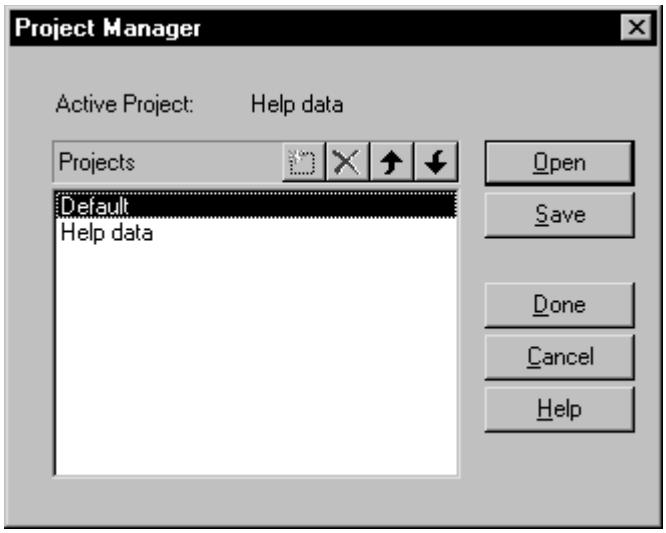
To edit a recorded macro

Open the text file that was saved as a macro.
Add any additional programming logic you wish to include.
Save the edited macro.

Project Configuration

The **Project Menu** offers an environment for working with projects.

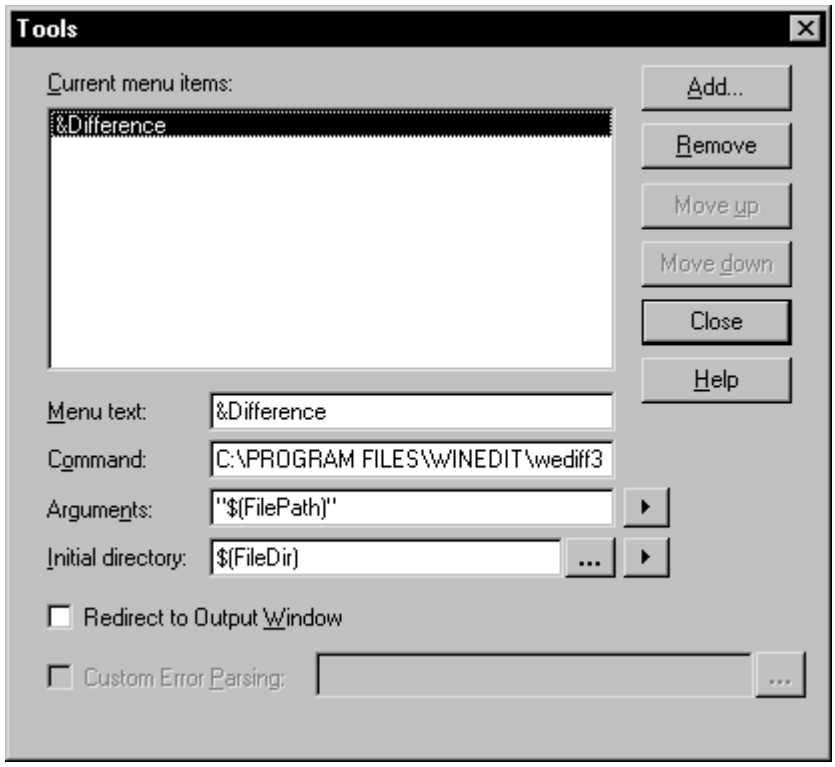
The Project Manager creates and saves projects which remember all your tools and open windows. Your projects will automatically be added as items to the bottom of the Project Menu allowing for quick changes from project to project.



Programs can be added to the Project Menu either with the **Customize Tools** / **Add** button or by recording and saving a macro.

In this dialog, Difference is added as an example.

Click on a dialog box item for more information.



Macro menu commands

See Also:
Debug macro

Context Menu

The Macro Menu is a completely configurable menu. It is here for your use. You can make it hop, jump and skip through your code in any way, shape or fashion you desire. The Macro menu dubs you master of the WinEdit universe and gives you the power through the Windows Interface Language to make your scripting tool fantasies come true.

As the Macro menu is completely configurable, we are not going to discuss the handy items we've included as samples of the power at your fingertips. Instead, we're nudging you in the right direction and giving you the do's and don'ts of macro menu configuration.

Customize this menu

The last option on the Macro Menu is Customize this menu. Selecting this item opens the Winedit.mnu file where all the code for this menu is kept. The language used to configure the menu is the Windows Interface Language. If you're not familiar with using the WIL language, you can find out more about it at "What is WIL?".

In order to place your items successfully onto the Macro menu, you must follow a few guidelines.

- Menus can be up to four levels deep.
- Levels are determined by the position of the first letter in the menu title.
- The top level menu starts at Column 1, the second starts in Column 2, and so on. The WIL code must begin at Column 5 or greater.
- Same level menu items must be separated by WIL code.

That's it. Those are the simplified rules. However, if you need more info, don't despair, everything you could ever want to know about writing Menu files can be found under this topic Menu file structure.

Good luck, and remember, "Use your power wisely".

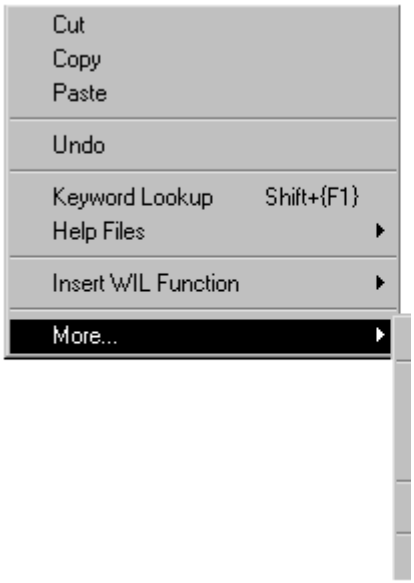
Context Menu

See Also:

Macro Menu Commands

Menu file structure.

WinEdit has a completely configurable context menu accessed by clicking the right mouse button anywhere within an open file. Right clicking results in a context menu drop down list filled with many useful macros. Using the Windows Interface Language, you can write your own macros and place them on this menu for easy access.



To make changes to the context menu, open the WEPOPUP.MNU file with **File/Open**, or access it from the context menu itself. Right click in the file, from the context menu dialog box select **More / How do I? / Customize this menu**.

Window menu commands

The Window menu offers the following commands, which enable you to arrange multiple views of multiple documents in the application window:

New Window	Creates a new window that views the same document.
Split	Split the active window into panes.
Cascade	Arranges windows in an overlapped fashion.
Tile	Arranges windows in non-overlapped tiles.
Arrange Icons	Arranges icons of closed windows.
Close	Closes the current window.
Close All	Closes all open windows.
Window manager	Lists all open windows. Multiple selections may be made.
Window 1, 2, ...	Goes to specified window.

Help menu commands

The Help menu offers the following commands, which provide you assistance with this application:

Help Topics	Offers you an index to topics on which you can get help.
About	Displays the version number of this application.

Command Line Options

WinEdit supports the following command line options:

Wildcard filenames:	WinEdit *.c
Multiple filenames	WinEdit fileone.txt filetwo.txt *.c
Line number:	WinEdit filename.txt -# 25
Macro:	WinEdit -M macroname.wbt
Print:	WinEdit -P filename.txt
PrintTo:	WinEdit -PT filename.txt printer driver port

Note: A space is required between the command line option and its parameters.

Debugging WIL Scripts

See Also:
Debug macro

Macro Menu Commands

WinEdit is used to both edit and debug scripts.

There are various ways to debug a script.

Experienced programmers who can write relatively bug-free code might just write a script and then hit the "Go" button and see if it works. If an error occurs, the line causing the error on it will be indicated and the current state of all variables displayed.

The rest of us will find that debugging a script is a more interesting process. We might write a few lines of code and then press the "**Step Into**" button to step through the code one line at a time. As each line is executed the current state of all variables is displayed in a special "**watch window**".

Once large sections of code are bug free, it becomes rather boring to step, step, step through each statement. There are several different solutions to this problem. One is the "**Step Over**" button that can be used to execute entire sections of code in a **GoSub** or **Call** function. In addition there is a "**Run to Cursor**" hot-key combination (Ctrl-F10) and menu item that allows you to place the mouse cursor on a line and execute to that point. In this way, it is possible to avoid step, step, stepping through large blocks of code.

For serious debugging there are "breakpoints". Breakpoints are useful where there is a large quantity of code and you are interested in debugging a specific section of it. To use breakpoints you click on a line of code and hit the "**Insert/Remove Breakpoint**" button. A red square will appear next to the line indicating a breakpoint is active on the line. Next you would click the "Go" button. The script will start executing and will stop when it hits a line with a breakpoint. You may have several different lines with breakpoints.

The **Watch Window** allows you to view the contents of script variables. From the View Menu select the "Watch" menu option to display the Watch Window at the bottom of the document area. Step through a script that contains several variables and observe the changing values. Notice that the latest variable assignment is displayed as the first line of text in the window.

Not only can you observe variable values, you can also change them. To make a change double-click on the variable name in the Watch Window, type a new value and click the OK button in the dialog that appears. Continue stepping to see the effect of your change.

Why change a value? It may not be necessary in simple scripts. However, as your scripts become more complex the relationships

between values and results may not be obvious. The suspected cause of a bug can quickly be verified or eliminated by placing a known good or bad value in a variable.

Also, the best time to test a block of script is right after you write it. When will you know it better? By setting variables in the debugger you can determine how your code fragment or subroutine will behave when it encounters normal, extreme and out of range values. This can be accomplished without having to write throwaway test driver scripts. You will be much more confident in your final script, if you know that each element is doing its job.

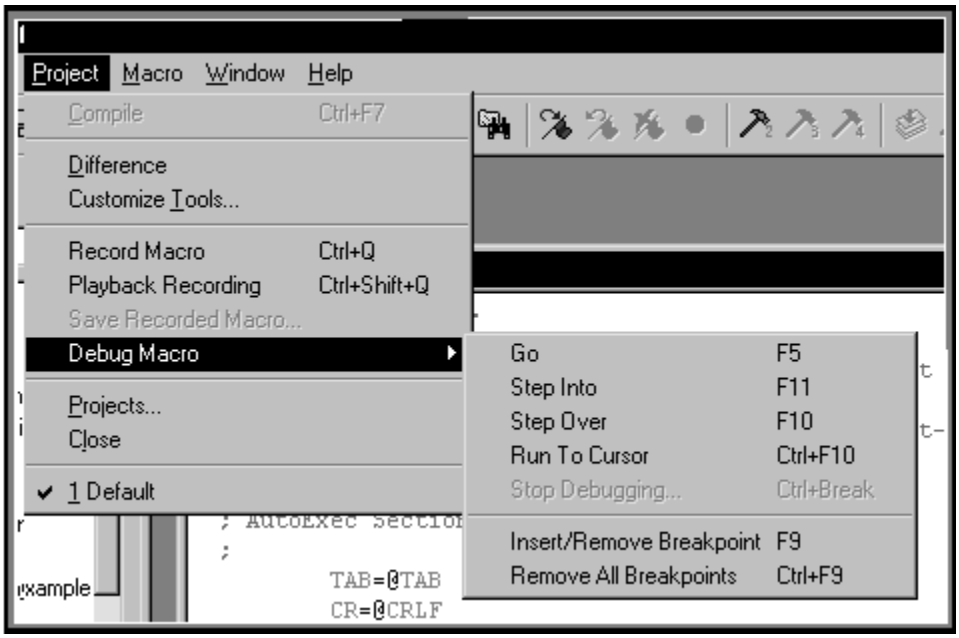
Debug Macro

WinEdit offers a complete interactive debugging environment for WIL script files. To debug a WIL script, first make sure the saved file is loaded and is the active document.

NOTE: The Debug commands will not be enabled unless the current document is defined as a **WIL file type**.

See Also:
Debugging WIL Scripts
Macro Menu Commands

Click on a dialog box item for more information or scroll down for a complete list of option descriptions.



Go	Begins executing the script commands. Execution will continue to the end of the script or until a breakpoint is encountered.
Step Into	Executes the current line of the script. If the current line is a goto, gosub, or call command, execution stops at the first line of the goto, gosub, or call code.
Step Over	Executes the current line of the script. If the current line is a goto, gosub, or call command, all the code at the goto, gosub, or call location is also executed.
Run To Cursor	Begins executing script commands at the current location and continues to the point in the script where the cursor (caret) is located.
Stop Debugging	Stops execution of the script.

Insert/Remove Breakpoint	Inserts a breakpoint at the current line, or removes it if it already exists. When execution of the script is initiated with the Go or Run To Cursor commands, execution will still stop if a line with a breakpoint is encountered.
Remove All Breakpoints	Removes all defined breakpoints in the current script.

Toolbars

WinEdit has four configurable toolbars; **File, Window, Debug** and **Tools**.

File	The File bar displays buttons for the typical File Management commands; Open, Save, Print etc.
Window	The Window bar allows for easy selection of window viewing options; Tiled; Cascade etc.
Debug	The Debug bar displays buttons for the WIL script debugging commands
Tools	The Tools bar displays programmable buttons for user defined tools.

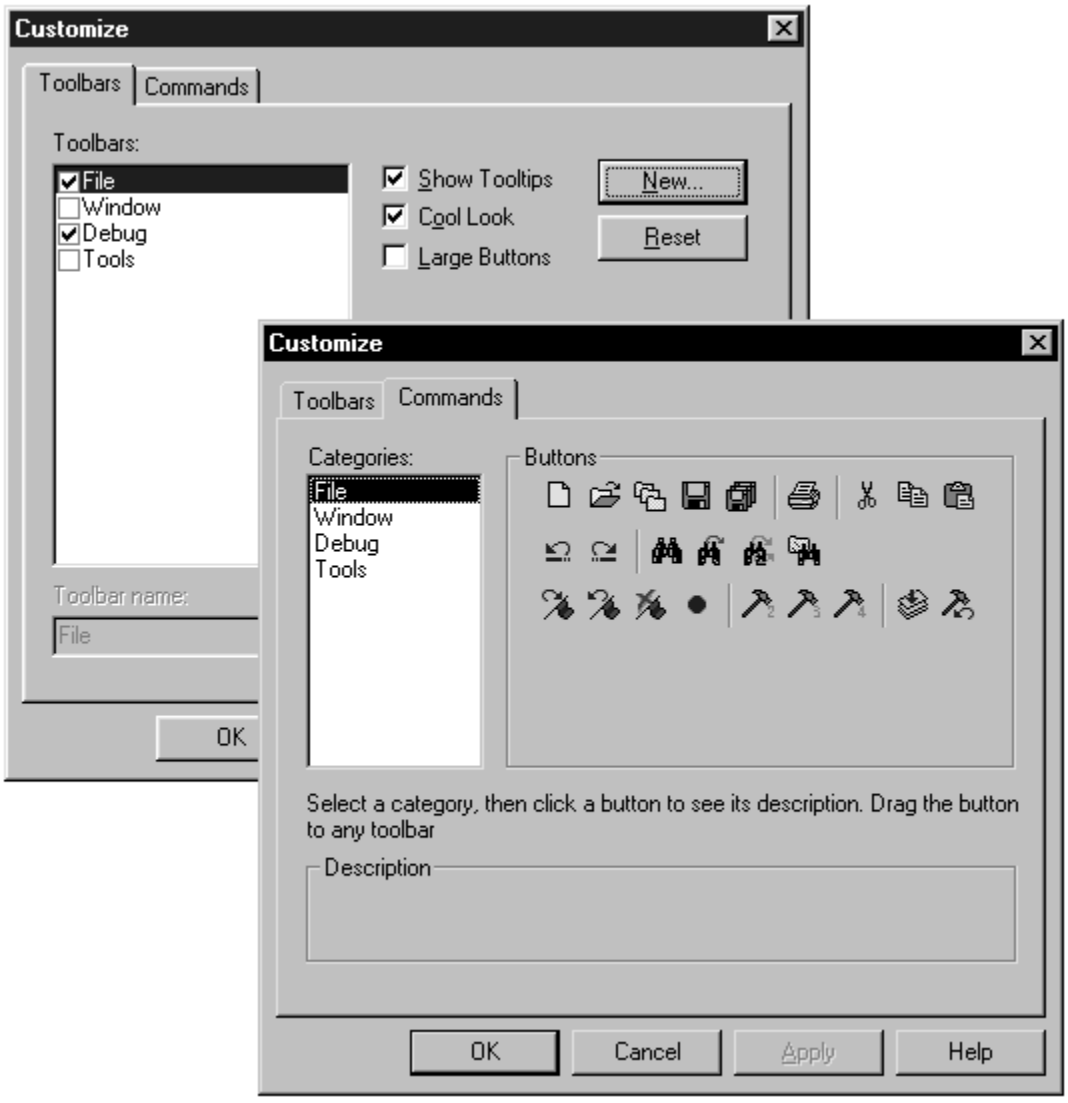
See Also:

- Toolbar Dialog Box
- Debug Macro
- Print Preview Toolbar

Toolbars dialog box

The **View / Toolbars** dialog box offers the following paged dialog boxes:

Toolbar	Place a check mark by the toolbars you wish to display. Click New to add additional toolbars you can create. Highlight a standard toolbar and click Reset to restore the toolbar to its original state.
Commands	Shows all available toolbar buttons for each standard toolbar. To customize toolbars, drag a button to or from this dialog box to any toolbar on display.



Options dialog box

Preferences for WinEdit are set from the **Options View** menu in one of the following paged dialog boxes.

Editor	Sets general options for WinEdit.
File type	Sets options for specific file types.
Keyboard	Sets keyboard shortcuts for menu commands.

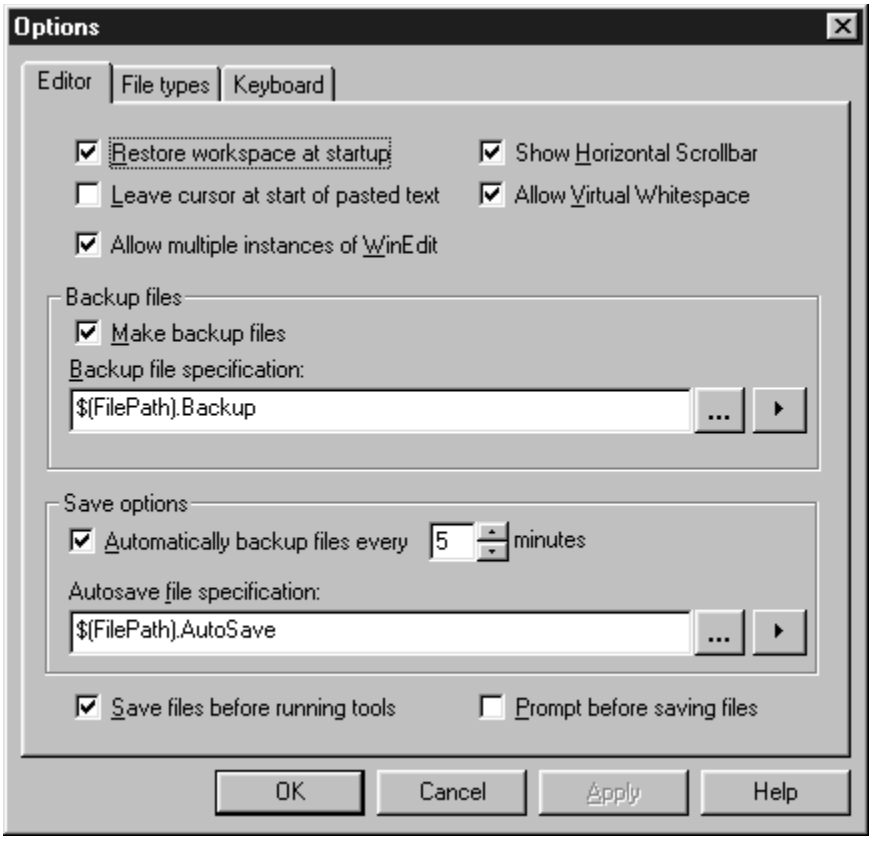
Special Help

- **Background File Checking**
- **Null Characters**

Options - Editor tab

The **View / Options Editor** dialog page offers the following items:

Click on a dialog box item for more information or scroll down for a complete list of option descriptions.



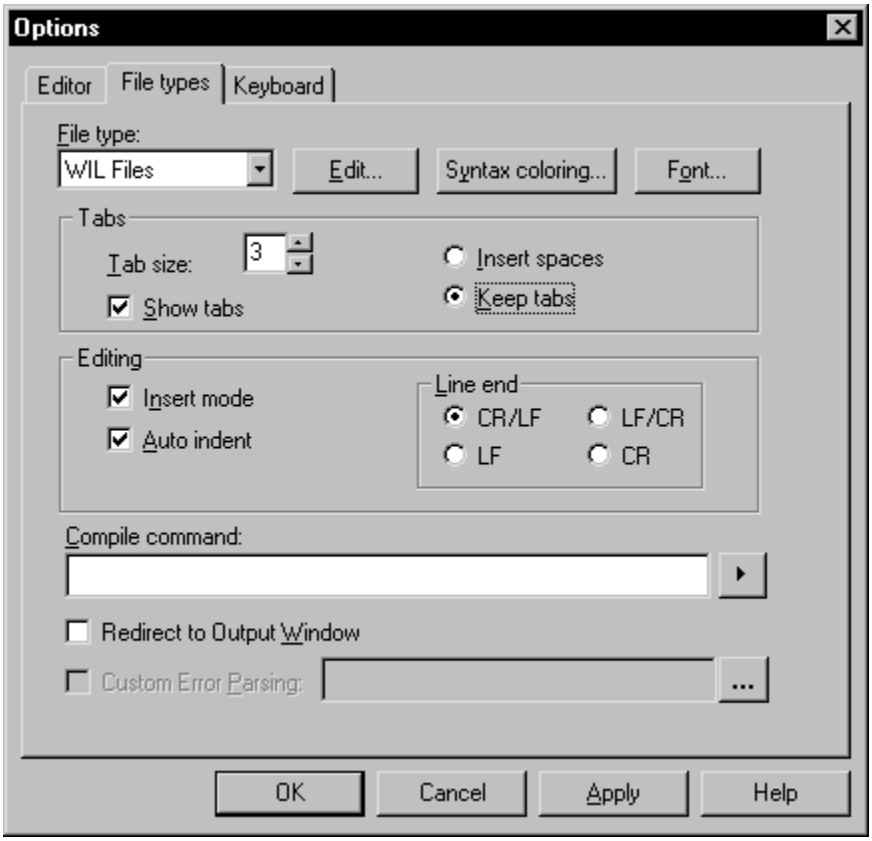
Restore workspace at startup	Reload all documents that were open at the end of the last editing session.
Leave cursor at start of pasted text	When checked, the cursor (caret) is positioned at the beginning of the pasted text. When unchecked, it is placed at the end of the pasted text.
Allow multiple instances of WinEdit	When unchecked, double clicking an associated file or starting WinEdit itself activates the already running instance instead of launching an additional copy.
Show horizontal scrollbar	When unchecked, no horizontal scrollbar is

	shown.
Allow virtual whitespace	When checked, the caret can be positioned in any column. When unchecked, the caret cannot be moved beyond the end of the text of any line.
Make backup files	When checked, a backup copy of a document is made whenever the document is saved.
Backup specification	Create a file specification to use when naming a backup file. Choose options based on the original document name from the menu button.
Automatically backup files	When checked, a backup copy of a document is made automatically at the time interval selected.
Autosave file specification	Create a file specification to use when naming an autosave file. Choose options based on the original document name from the menu button.

Options - File types tab

In WinEdit, many editing options are associated with a particular file type. WinEdit classifies file types based upon the file extension.

Click on a dialog box item for more information or scroll down for a complete list of option descriptions.

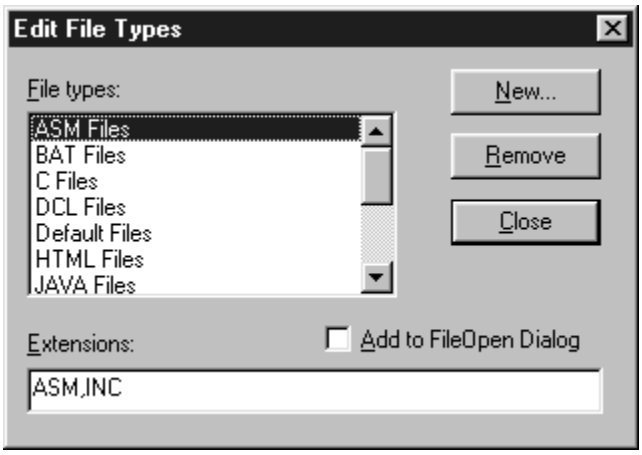


File type	Chooses a file type from the dropdown list.
Edit	Allows adding or deleting of file types from the list.
Syntax coloring	Sets options for coloring keywords for this file type, and for specifying the characters which flag text as a 'comment'.
Font	Selects a fixed pitch font to be used to display this file type.
Tab size	Sets the number of columns each tab character represents.
Show tabs	When selected, tab characters are displayed on

	screen.
Insert spaces/Keep tabs	Selects whether to insert a tab character, or a corresponding number of space characters, when pressing the tab key.
Insert mode	When checked, text typed is inserted at the caret position. When unchecked, overtype mode is used, where text typed replaces text at the caret position.
Line End	Selects the characters that are inserted when the Enter key is pressed.
Compile command	Enter the command line used to compile this file type. Command line parameters based upon the original document name can be selected from the menu button.
Redirect to output window	Captures command line output and displays it in the output window when this file type is compiled from within WinEdit. Double click on an error or warning or choose Next Error/Previous Error from the Search menu to move to that position in your source code file.
Custom error parsing	If the compiler output from this file type does not match the format "filename(lineno):text", you can process the output in a WIL script files so that Next Error/Previous Error will work correctly. The sample script mserror.wbt demonstrates the required processing.

Edit File Types

Allows adding or deleting of file types from the list.



Edit existing or Add new file extensions in the Extensions edit box. Delimit extension lists with a comma.

Options - Keyboard tab

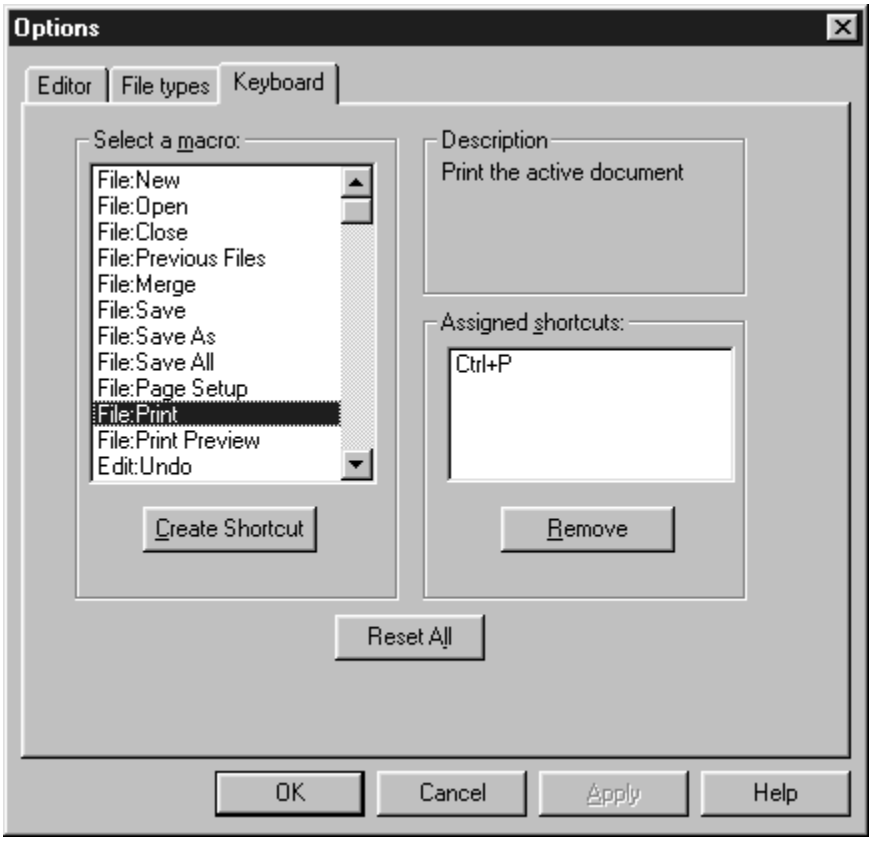
The **View / Options Keyboard** dialog box allows you to choose the accelerator keys for any WinEdit command.

Choose a command from the list, and then click **Create Shortcut** to assign a keystroke shortcut.

To **Remove** an existing shortcut, select the command name then highlight the existing accelerator key in the second list and click Remove.

In order to change accelerator keys, first remove any existing keystroke then use Create Shortcut to reassign the command.

For a list of default key mappings, see WinEdit Key Mappings.

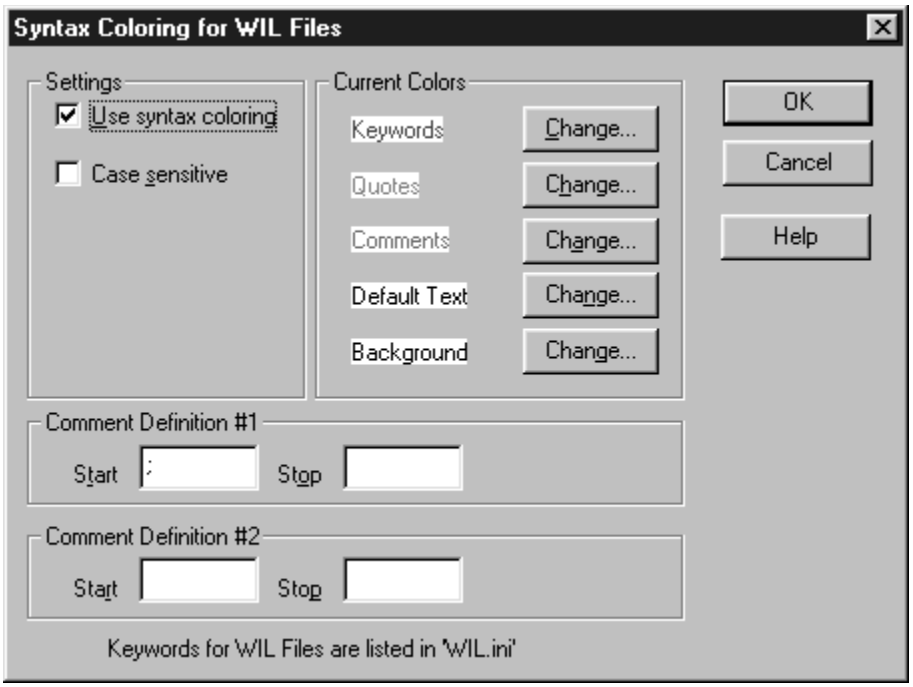


Syntax coloring

Syntax Coloring sets options for coloring keywords and characters which comment text for the selected file type. This option can be accessed through the **View / Options File types** dialog box.

See Also:
Color Files

WinEdit comes Pre-loaded with keywords for the fifteen most common languages. However, there is no limit. See the instructions in **Color Files** for information on creating your own keyword files.



WinEdit Help File

Color Files

See Also:
Syntax Coloring

Pick your colors

WinEdit has the capability to color highlight Keywords, Comments and String Literals for any language file. After defining a File type from the **View Menu / Options File types** dialog box, colors can be chosen for the selected file type using the **Syntax Coloring** button to access the Syntax Coloring dialog box.

Set your Keywords

Once you've established colors for your file, you then need to define your keywords in a language specific initialization file with an extension of *.CLR* WinEdit is pre-loaded with keyword *.CLR* files for the 15 most common languages. If your language is not pre-loaded, you can easily create a *.CLR* file by either copying an existing file and making the necessary changes or by opening a new file and starting from scratch.

Characteristics of a valid language.CLR file:

- The name of the file is the same name which appears in the File Types list box. As an example, an entry of WIL Files in the **View Menu / Options File types** dialog box will correspond to a .clr filename of WIL.clr in the WinEdit directory.

- Any File extension listed under a "file type" will use the corresponding clr file for its color highlighting. The File type list box shows extensions of WIL, WBT, MNU, MNW, and MAC under the entry WIL files. The file WIL.clr will provide color highlighting for all of these extensions.

- All language CLR files must be comprised of two sections; [COLORS] and [KEYWORDS].

  In the [COLORS] section you can define additional colors to be used for color highlighting. This allows color changes to be made to certain file types or specific keywords types independent of the default color specified in the Syntax Coloring dialog box.

  ```
  [COLORS]
  WBT=128,0,128
  MNU=255,0,255
  ```

  Your Keywords need to be listed in the [KEYWORDS] section. As you list your keywords, you can specify that additional colors override the default color previously established in the Syntax Coloring dialog box.

  ```
  [KEYWORDS]
  else=1                (value of 1 uses the default color.)
  ```

end=WBT	(uses WBT color set up in [COLORS] section.)
endif=MNU	(uses MNU color set up in [COLORS] section.)

Font Selection

The font which appears on the screen is not necessarily the font used when you print. Print and Screen Fonts are established using separate Font Dialog boxes.

Print Font

From the **File** menu **Page Setup** command, use the radio button to emulate the Screen Font or select the Printer Font button to change your Printer font.

Screen Font

From the **View** menu **Options File type** command, the Font button selects a fixed pitch font to be used to display the selected file type.

Using Regular Expressions

A regular expression is a search or replace string that uses special characters to match text patterns. WinEdit supports UNIX style regular expressions.

When WinEdit conducts a search using regular expressions, it must check character by character in your text. For this reason, searches using regular expressions are slower than regular searches.

The following table describes the regular expression characters recognized by WinEdit.

Expression	Description
****	*Escape*. WinEdit will ignore any special meaning of the character that follows the Escape expression. Use the Escape if you need to search for a literal character that matches a regular expression character.
.	*Wild Card*. Matches any character. For example, the expression 'X.X' will match 'XaX', 'XbX, and 'XcX', but not 'XaaX'.
^	*Beginning Of Line*. The expression matches only if it occurs at the beginning of a line. For example, '^for' matches the text 'for' only when it occurs at the beginning of a line.
$	*End Of Line*. The expression matches only if it occurs at the end of a line. For example, '(void)$' matches the text '(void)' only when it occurs at the end of a line.
[ ]	*Character Class*. The expression matches any character in the class specified within the brackets. Use a dash (-) to specify a range of character values. For example, '[a-zA-Z0-9]' matches any letter or number, and '[xyz]' matches 'x', 'y', or 'z'.
[^]	*Inverse Class*. The expression matches any character not specified in the class. For example, '[^a-zA-Z]' matches any character that is not a letter.
*****	*Repeat Operator*. Matches zero or more occurrences of the character that precedes the '*'. For example, 'XY*X' matches 'XX', 'XYX', and 'XYYX.
+	*Repeat Operator*. Matches one or more occurrences of the character that precedes the '+'. For example, 'XY+X' matches 'XYX' and 'XYYX, but not 'XX'.

mserror.wbt

This is a sample script that shows the processing necessary to enable error parsing In WinEdit, if the compiler output does not match “Microsoft style” compiler output (“filename(lineno):text”).

```
;---------------------------------------------------------------------;
; mserror.wbt                                                         ;
;                                                                     ;
; Error parsing for Microsoft style output:                           ;
; "filename(lineno):text"                                             ;
;                                                                     ;
; This script is included only as a template for you to use in        ;
; creating a custom error parsing script, since Microsoft style       ;
; output is handled internally by WinEdit.                            ;
;                                                                     ;
; 1.  If called with param1 == "@GetErrorWords", return               ;
;     a comma-delimited list of words which identify a line           ;
;     as containing an error. In Microsoft style output that          ;
;     would be "error,warning".                                       ;
;                                                                     ;
; 2.  Otherwise, parse the variable @strLine to extract the filename ;
;     and line number.  In this case, the filename is all the         ;
;     characters up to the opening parenthesis, and the line number   ;
;     is all of the following characters up to the closing            ;
;     parenthesis.                                                    ;
;                                                                     ;
;---------------------------------------------------------------------;

@ErrFile = ""   ; return value
@ErrLine = 0    ; return value
@ErrWords = ""  ; return value

;---------------------------------------------------------------------;
;  Return the error words                                             ;
;---------------------------------------------------------------------;
if param1 == "@GetErrorWords"
@ErrWords = "error,warning"
exit
endif

;---------------------------------------------------------------------;
; extract the file name by searching for the '(' character           ;
;---------------------------------------------------------------------;
finish = StrScan(@strLine, "(", 0, @FWDSCAN)
if finish == 0
exit
endif
strFile = strsub(@strLine,1,finish-1)
if (FileExist(strFile))
@ErrFile = strFile
else
drop(strFile,finish)
exit
endif
```

```
;-------------------------------------------------------------------------;
; extract the line number, which follows the '(' character                ;
;-------------------------------------------------------------------------;
start = finish+1
finish = StrScan(@strLine, ")", start, @FWDSCAN)
if finish == 0
@ErrFile = ""
drop(strFile,start,finish)
exit
endif
strLine = StrSub(@strLine,start,finish-start)
if IsInt(strLine)
@ErrLine = strLine
else
@ErrFile = ""
endif

drop(strLine,strFile,start,finish)
```

WinEdit Help File

WinEdit Command Reference

See Also:
Basic Box Functions

The following script commands are specific to WinEdit. In addition to these commands, any WIL command may be used in a script.

Editing

wCopy()
wCopyLine()
wCopyAppend()
wCopyMarked()
wCut()
wCutLine()
wDelete()
wInsString(string)
wNewLine()
wPaste()
wBackspace()
wInvertCase()
wUpperCase()
wLowerCase()
wRedo
wUndo()

Text Selection

wClearSel()
wSelUp()
wSelDown()
wSelLeft()
wSelRight()
wSelEnd()
wSelHome()
wSelPgUp()
wSelPgDn()
wSelWordLeft()
wSelWordRight()
wSelTop()
wSelBottom()
wSelectAll()
wSetColBlk()
wStartSel()
wEndSel()
wSelInfo()

Cursor Movement

wHome()
wEnd()
wTopOfFile()
wEndOfFile()
wUpLine()
wDownLine()
wLeft()
wRight()
wPageUp()
wPageDown()
wWordLeft()
wWordRight()
wTab()
wBackTab()
wToggleIns()
wSetBookmark()

File

wFileNew()
wFileOpen(filename)
wFileMerge(filename)
wFileRevert()
wFileSave()
wFileSaveAs(filename)
wFilePgSetup()
wFilePrint()
wPrintDirect()
wPrinSetup()
wProperties()
wFileExit()

Window

wWinArrIcons()
wWinCascade()
wWinClose()
wWinCloseAll()
wWinMaximize()
wWinMinimize()
wWinNext()
wWinRestore()
wWinTile()

Status

wGetChar()
wGetColNo()
wGetFileName()
wGetIns()
wGetLineNo()
wGetModified()
wGetRedo()
wGetSelState()
wGetUndo()
wGetWord()
wGoToLine(lineno)
wGoToCol(colno)

Compiling

wCompile()
wNextError()
wPrevError()
wSetProject(ProjectName)

Searching

wFind
wRepeat()
wChange()

Miscellaneous

wStatusMsg(message)
wViewOptions()
wViewOutput()
wGetOutput()
wLineCount()

Internet

wFTPOpen()
wFTPSave()
wViewHTML()

Additional Macro Commands

Graphical box drawing functions have been added to WinEdit They're nothing fancy but they get the job done.

Basic boxes functions :

BoxOpen(title, text) Opens a WinBatch message box.
BoxShut( ) Closes the WinBatch message box.
BoxText(text) Changes the text in the WinBatch message box.
BoxTitle(title) Changes the title of the WinBatch message box.

Additional Box functions are:

BoxButtonDraw(box ID, button ID, text, coordinates)
BoxButtonKill(box ID, button ID)
BoxButtonStat(box ID, button ID)
BoxButtonWait( )BoxCaption(box ID, caption)
BoxColor(box ID, color, wash color)
BoxDestroy(box ID)
BoxDrawCircle(box ID, coordinates, style)
BoxDrawLine(box ID, coordinates)
BoxDrawRect(box ID, coordinates, style)
BoxDrawText(box ID,coordinates,text,erase flag,alignment)
BoxesUp(coordinates, show mode)
BoxMapMode(box ID, map mode)
BoxNew(box ID, coordinates, style)
BoxPen(box ID, color, width)
BoxTextColor(box ID, color)
BoxTextFont(box ID, name, size, style, pitch & family)
BoxUpdates(box ID, update flag)

More Useful functions:

BreakPoint Causes a breakpoint on the next statement when used with a script debugger. Otherwise the command does nothing.
intcontrol(...) (see WIL manual)
reload
rtstatus
version() New function that returns the Winedit99+ version number

wFileNew()

Comments

wFileNew creates a new MDI child window.

Example:

```
wSelectAll()
wCopy()
wFileNew()
wPaste()
```

The above commands will copy the contents of the active document window and paste the contents of the window into a new document window.

wFileOpen(filename)

Comments:

wFileOpen creates a new MDI child window and reads an existing file into the window. To open a file without prompting, pass a valid file name to wFileOpen. If the FileName parameter is "", the File Open dialog box will appear prompting the user for a filename.

Example:

```
wFileOpen("")
```

The above command will prompt the user for a filename to open. To open a file directly without prompting, use the following syntax:

```
wFileOpen("FILENAME.TXT")
```

wFileMerge(filename)

Comments:

wFileMerge reads an existing file into the active MDI child window. To merge a file without prompting, pass a valid file name to wFileMerge in the FileName parameter. If FileName is "", the File Merge dialog box will be used to obtain a file name from the user.

Example:

```
wFileMerge("")
```

The above command will prompt the user for a filename to merge. To merge in a file directly without prompting, use the following syntax:

```
wFileMerge("FILENAME.TXT")
```

The indicated file is merged at the insertion position in the active document window.

wFileRevert()

Comments:

wFileRevert rereads the current document from disk. Use this command when you wish to discard all changes you have made to a document.

wFileSave()

Comments:

wFileSave saves the file in the currently active MDI child window without prompting (same as selecting Save from the File menu).

wFileSaveAs(filename)

Comments:

wFileSaveAs saves the file in the currently active MDI child window to a new filename.

Example:

```
wFileSaveAs("")
```

The above command will prompt the user for a filename. To save the file directly to new file name without prompting, use the following syntax:

```
wFileSaveAs("FILENAME.TXT")
```

wFilePrint()

Comments:

wFilePrint prints the text in the currently active MDI child window (same as choose Print from the File menu).

wFilePgSetup()

Comments:

wFilePgSetup brings up the Page Setup dialog box (same as choosing Page Setup from the File menu).

wPrinSetup()

Comments:

wPrinSetup brings up a dialog box listing all installed printers (same as selecting Printer Setup from the File menu). The user can choose a printer from the list and WinEdit will use the selected driver for all print jobs. The user can also access the printer driver setup dialog by choosing the Setup button.

wFileExit()

Comments:

Command to exit WinEdit. If there are any unsaved files, the user will be prompted to save before closing. The user can cancel the exit operation at that point. If there are no unsaved files, the exit is automatics (no chance to cancel the exit).

wFind(SearchText,Forward,MatchCase,Regex,Wrap)

Comments:

wFind searches for the text identified by SearchText parameter. If Forward is TRUE, the search direction is forward. If MatchCase is TRUE, then the search is case sensitive. If Regex is TRUE, then regular expressions are used. If Wrap is TRUE, the entire file will be searched from the current position, to the end of the file, and then continuing at the beggining of the file, if not found.

Example:

```
wFind("Blue",1,1,0,1)
```

The above example searches forward through the document window for the word Blue.

wGetChar()

Return Value

Returns the character to the right of the insertion point.

Example:

```
a=wGetChar()
wInsString(a)
```

This example gets the character to the right of the insertion point and inserts the character into the document window.

wGetFileName()

Comments:

wGetFileName returns a string with the fully qualified path name of the active MDI child window.

Example:

```
a=wGetFileName()
wInsString(a)
```

This example gets the filename for the active document window and inserts the filename (with the path information) at the insertion point.

wGetIns()

Return Value

Returns TRUE (1) if Insert is on, FALSE (0) if Overtype is on.

Example:

```
a=wGetIns()
If a == 0 Then Message ("Title", "Overtype is on")
If a == 1 Then Message ("Title", "Insert Mode is on")
```

The above commands assign the return value of wGetIns() to the "a" variable and then test for whether "a" is True or False. The If command used above to evaluate the "a" variable is a WIL (Windows Interface Language) command. Look to the WIL.HLP file for more information on the WIL commands.

wGetSelState()

Return Value

The result is TRUE if there is a selection, otherwise the function returns zero.

Example:

```
a=wGetSelState()
If a == 1 Then wCopy()
```

This example checks whether there is a selection, and if True copies the selection to the clipboard.

wGetRedo()

Return Value

The result is TRUE (1) if any operation can be redone. Otherwise wGetRedo returns zero.

Example:

```
a=wGetRedo()
If a == 1 Then wRedo()
```

The above example checks whether the last edit can be redone and if the return value is TRUE, the edit if redone ("wRedo()" is the same as choosing Redo from the Edit menu).

wGetUndo()

Return Value

The result is TRUE (1) if any operation can be undone. Otherwise wGetUndo returns zero. ("wUndo()" is the same as choosing Undo from the Edit menu).

Example:

```
a=wGetUndo()
If a == 1 Then wUndo()
```

The above example checks whether the last edit can be undone and if the return value is TRUE, the edit if undone.

wGetColNo()

Return Value

Returns the column number position for the insertion position in the active MDI child window. wGetColNo returns 0 if unsuccessful.

Example:

```
a=wGetColNo()
Message("Column Number", a)
```

The above commands get the column number for the insertion point and post the results in a message box. Look to the WIL.HLP file for more information on WIL commands such as the Message command.

wGetLineNo()

Return Value

Returns the line number position for the insertion position in the active MDI child window. wGetLineNo returns0 if unsuccessful.

Example:

```
a=wGetLineNo()
Message("Line Number", a)
```

The above commands get the line number for the insertion point and post the results in a message box. Look to the WIL.HLP file for more information on WIL commands such as the Message command.

wGetModified()

Return Value

TRUE if the active MDI child has been modified.

Example:

```
a=wGetModified()
If a == 1 Then Message ("Mod", "Text has been modified")
```

The above example will post a message if the text in the document window has been modified.

wNextError()

Comments:

wNextError displays the next warning or error message on the status line.

wPrevError()

Comments:

wPrevError displays the previous warning or error message on the status line.

wRepeat()

Comments:

wRepeat conducts a search using the same search string used in the previous search.

Example:

```
wFind("Blue",1,1)
PlayWaveForm("tada.wav", 0)
wRepeat()
```

This example searches forward for the word Blue, plays the TADA.WAV file and then repeats the wFind statement. The PlayWaveForm command used above is a WIL (Windows Interface Language) command. Look to the WIL.HLP file for more information on the WIL commands.

wSetProject(ProjectName)

Comments:

wSetProject sets the current project to ProjectName, without bringing up the Project Management dialog box.

wCompile()

Comments:

wCompile executes the Compile command syntax entered in the **View / Options File types** dialog box.

wStatusMsg(message)

Comments:

wStatusMsg() displays the string "message" on the WinEdit status line.

wViewOutput()

Comments:

wViewOutput() shows the Output window, where compiler output or Find In Files results are displayed.

wWinArrIcons()

Comments:

wWinArrIcons rearranges all minimized MDI child windows icons along the bottom of the WinEdit application window.

Example:

```
wFileOpen("accel.rc")
wWinMinimize()
wFileNew()
wWinMinimize()
wWinArrIcons()
```

The above example opens the ACCEL.RC file and a new document window, minimizes them both and then arranges the icons left to right along the bottom of the WinEdit application window.

wWinCascade()

Comments:

wWinCascade cascades all MDI child windows (arranges all of the open windows in a stack).

wWinClose()

Comments:

wWinClose closes the active MDI child window. If there are unsaved changes, the user is prompted to save the changes before the file is closed.

wWinCloseAll()

Comments:

wWinCloseAll closes all MDI child windows. If there are unsaved changes, the user is prompted to save the changes to each file before the file is closed.

wWinMaximize()

Comments:

wWinMaximize maximizes the active MDI child window.

Example:

```
wFileNew()
wWinMaximize()
```

This example opens a new document window and maximizes the window.

wWinMinimize()

Comments:

wWinMinimize minimizes the active MDI child window to an icon at the bottom of the WinEdit application window.

Example:

```
wFileOpen("accel.rc")
wWinMaximize()
```

This example opens the ACCEL.RC file and minimizes the window to an icon.

wWinNext()

Comments:

wWinNext brings the focus to the next MDI child window.

wWinRestore()

Comments:

wWinRestore restores the active MDI child window to its non-minimized, non-maximized state.

wWinTile()

Comments:

wWinTile tiles all MDI child windows. If there are three or less windows, the windows will be tiled horizontally left to right.

wChange()

Comments:

wChange brings up the Replace dialog box.

wBackspace()

Comments:

wBackSpace deletes the character to the left of the current position. This command is the equivalent of pressing the backspace character on the keyboard.

Example:

```
wBackSpace()
wHome()
```

The above example deletes the character to the left of the cursor and moves the cursor to the beginning of the line.

wCopy()

Comments:

wCopy copies the selected text to the Windows clipboard.

Example:

```
wSelWordLeft()
wCopy()
```

The above commands will select the word to the left of the cursor and copy it to the Windows clipboard.

wPaste()

Comments:

wPaste pastes text from the clipboard into the active WinEdit document window.

Example:

```
wSelectAll()
wCopy()
wFileNew()
wPaste()
```

The above commands will copy the contents of the active document window and paste the contents of the window into a new document window.

wCopyLine()

Comments:

wCopyLine copies the current line to the clipboard if there is no selection. If there is a selection, wCopyLine calls wCopy and copies the selected text to the clipboard.

Example:

```
wCopyLine()
wDownLine()
wPaste()
```

The above example copies the line of text where the cursor resides, moves down a line, and pastes the line of text from the clipboard.

wCut()

Comments:

wCut cuts the current selection to the clipboard. The text cut to the clipboard can be later inserted into a document with the wPaste command.

See Also:

wDelete
wPaste

wCutLine()

Comments:

wCutLine cuts the current line to the clipboard if there is no selection. If text is selected, then wCutLine calls wCut and cuts the selected text to the clipboard.

Example:

```
wCutLine()
wGoToLine(4)
wPaste()
```

The above example cuts the contents of the current line to the clipboard and pastes the line on line 4 of the active document.

wDelete()

Comments:

wDelete deletes either the current selection or, if there is no selection, the character following the current position without copying the text to the clipboard. This command is the equivalent of pressing the Del or Delete character on the keyboard.

Example:

```
wDelete()
wHome()
```

The above example deletes the character to the right of the cursor and moves the cursor to the beginning of the line.

See Also:

wCut

wGoToLine(lineno)

Comments:

wGoToLine moves the current position to the line number identified by the lineno parameter. If the line number is greater than the last line in the file, the current position is moved to the last line in the file.

Example:

```
wGoToLine(6)
```

The above command will move the cursor to line 6 in the document file while maintaining the current column position. So if your cursor is positioned on Line 13, Col 21, the cursor position will be Line 6, Col 21 after the above command is executed.

See Also:

wGoToCol

wGoToCol(colno)

Comments:

wGoToCol moves the current cursor position to the column identified by the colno parameter.

Example:

```
wGoToCol(10)
```

The above command will move the cursor to colum 10 in the document file while maintaining the current line position. So if your cursor is positioned on Line 13, Col 21, the cursor position will be Line 13, Col 10 after the above command is executed.

See Also:

wGoToLine

wHome()

Comments:

wHome moves the current cursor position to Column 1 (the beginning of the line).

Example:

```
wHome()
wPaste()
```

The above commands will move the cursor to the beginning of the line and paste in the contents of the clipboard.

wEnd()

Comments:

wEnd moves the cursor position to the column following the last text or space character in the current line.

Example:

```
wEnd()
wInsString("Hello")
```

The above commands will insert the text Hello at the end of the current line.

wTopOfFile()

Comments:

wTopOfFile moves the cursor position to Line 1, Column 1 (the equivalent of pressing CTRL+Home).

Example:

```
wTopOfFile()
wInsString("Top of File")
```

The above commands will insert the text "Top of File" at the beginning of the document window (Line 1 Column 1).

wEndOfFile()

Comments:

wEndOfFile moves the cursor position to the column following the last text character on the last line of the file (the equivalent of pressing CTRL+End).

Example:

```
wEndOfFile()
wInsString("End of File")
```

The above commands will insert the text "End of File" after the last text in the document window.

wUpLine()

Comments:

wUpLine moves the current cursor position to the previous line (moves to the line above the current line).

Example:

```
wUpLine()
wHome()
```

The above commands will move the cursor position to the beginning of the previous line.

wDownLine()

Comments:

wDownLine moves the current position to the next line (moves to the line below the current line).

Example:

```
wDownLine()
wEnd()
```

The above commands will move the cursor position to the end of the next line.

wLeft()

Comments:

wLeft moves the current position one column to the left. If the current position is Column 1, the current position is moved to the end of the previous line.

Example:

```
wLeft()
wTab()
```

The above commands will move the cursor position one position to the left and insert a tab.

wRight()

Comments:

wRight moves the current position one column to the right.

Example:

```
wRight()
wTab()
```

The above commands will move the cursor position one position to the right and insert a tab .

wPageUp()

Comments:

wPageUp moves the current position up one screenful of text (equivalent of pressing PgUp on the keyboard).

wPageDown()

Comments:

wPageDown moves the current position down one screenful of text (equivalent of pressing PgDn on the keyboard).

wWordLeft()

Comments:

wWordLeft moves the cursor position one word to the left (the cursor will be positioned just before the word to the left of the current cursor position).

wWordRight()

Comments:

wWordRight moves the current position one word to the right (the cursor will be positioned just before the word to the right of the current cursor position).

wTab()

Comments:

wTab inserts a number of spaces and moves the current position to the next tab stop. If more than one line is selected, every line within the selection is shifted to the right one tab stop.

wBackTab()

Comments:

wBackTab moves the current position to the previous tab stop. If there is a selection, every line within the selection is shifted to the left one tab stop.

wGetWord()

Comments:

wGetWord returns the word at the current cursor position. If the cursor is not on an alphanumeric character, an empty string is returned.

Example:

```
A=wGetWord()
Message("Title",A)  ; WIL Command, see WIL.HLP
```

The above commands get the word where the insertion point is positioned and assign the text to the variable "A". The Message command is used to display the contents of the A variable in a message box. The "Message" command is a WIL (Windows Interface Language) command. Look to the WIL.HLP file for more information on the WIL commands.

wSelectAll()

Comments:

wSelectAll selects all the text in the active document window. The insertion position is moved to the end of the file.

Example:

```
wSelectAll()
wCopy()
wFileNew()
wPaste()
```

The above commands will copy the contents of the active document window and paste the contents of the window into a new document window.

wInsString(string)

Comments:

wInsString inserts string at the current position.

Example:

```
A=wGetWord()
wDownLine()
wGoToCol(1)
wInsString(A)
```

The above commands get the word where the insertion point is positioned and assign the text to the variable "A". The remaining commands insert the contents of the A variable at the beginning of the next line.

wNewLine()

Comments:

wNewLine is equivalent to pressing the "Enter" key to break a line at the current position.

Example:

```
wGoToCol(10)
wNewLine()
```

The above commands move the current position to column 10 and inserts a new line at that point.

wSetColBlk()

Comments:

wSetColBlk enables column block marking for the next block operation. WinEdit automatically returns to stream block marking after the next block operation.

Example:

```
wSetColBlk()
wHome()
wSelEnd()
wSelDown()
wSelDown()
wCopy()
wHome()
wDownLine()
wPaste()
```

The first five lines above will block select all characters to the right of the insertion point on the current line and the two line below. Once marked, the text is copied to the clipboard and pasted at the beginning of the following line.

wToggleIns()

Comments:

wToggleIns toggles the insert state between Insert and Overtype modes (INS or OVR indicates the insert state on the status bar). If Insert Mode is selected under the Edit menu (turned "on"), then the wToggleIns() command will toggle to OverType mode.

See Also:

wGetIns

wRedo()

Comments:

Equivalent of selecting Redo from the Edit menu. The wRedo() command allows you to reverse any Undo command.

See Also:

wGetRedo

wUndo()

Comments:

Allows you to "undo" the most recent editing action.

See Also:

wGetUndo

wPrintDirect()

Comments:

wPrintDirect prints the current document without first bringing up the Print dialog box.

wProperties()

Comments:

wProperties brings up the Properties dialog box, which displays information about the current document.

wViewOptions()

Comments:

wViewOptions brings up the **View / Options** dialog box.

wSelInfo()

Comments:

wSelInfo returns information about the current selection (if any); It returns a TAB delimited list with the line number and column number of the beginning of the selection, and the line number and column number of the end of the selection.

Example:

```
wGoToLine(20)
wGoToCol(10)
wStartSel()
wGoToLine(40)
wGoToCol(20)
wEndSel()
list = wSelInfo()   ; returns a tab-delimited list of the block bounds
BlockStartLine = ItemExtract( 1, list, @TAB)
BlockStartCol  = ItemExtract( 2, list, @TAB)
BlockEndLine   = ItemExtract( 3, list, @TAB)
BlockEndCol    = ItemExtract( 4, list, @TAB)
Message("BlockInfo", "Block start: Line %BlockStartLine% Column
%BlockStartCol%%@CRLF%Block End:   Line %BlockEndLine% Column
%BlockEndCol%")
```

wLineCount()

Comments:

wLineCount returns the number of lines in the current document.

Example:

```
count = wLineCount()
Message("LineCount", "Total lines in this file: %count%")
```

wFTPOpen()

Comments:

wFTPOpen launches WinEdit's FTP application, **WE_FTP.EXE**, to retrieve and open a remote file.

wFTPSave()

Comments:

wFTPSave launches WinEdit's FTP application, **WE_FTP.EXE**, to allow you to save the currently open file on a remote system. WE_FTP will be passed the current file name, which will be displayed near the bottom of its screen. Navigate to the remote site and directory in which you wish to save the file, and press the "Save" button.

wViewHTML()

Comments:

wViewHTML opens the current file in WEXPLORER.EXE, WinEdit's HTML browser.

wInsLine(string)

Comments:

wInsLine inserts text into the current document, followed by a linefeed. It is equivalent to calling wInsString(string), followed by wNewLine().

wGetOutput()

Comments:

wGetOutput returns the text (if any) in the Output window. You can use this command if you need to parse a tool's output in ways that WinEdit is not designed to do.

Example:

```
string = wGetOutput()
Message("output window", "The output window contains this: %string%")
```

wStartSel()

Comments:

wStartSel inserts an ‘anchor’ at the current cursor position, which, when followed by a matching call to wEndSel() completes the marking of a selection. In between the two commands any cursor movement commands may be used.

wEndSel()

Comments:

wEndSel completes the marking of a selection that was begun with the wStartSel command. In between the two commands any cursor movement commands may be used.

wClearSel()

Comments:

wClearSel removes any selection from the text in the current document.

wSelUp()

Comments:

wSelUp begins selecting text if not currently selecting, and extends the selection up one line from the current position.

wSelDown()

Comments:

wSelDown begins selecting text if not currently selecting, and extends the selection down one line from the current position.

wSelLeft()

Comments:

wSelLeft begins selecting text if not currently selecting, and extends the selection left one character from the current position.

wSelRight()

Comments:

wSelRight begins selecting text if not currently selecting, and extends the selection right one character from the current position.

wSelEnd()

Comments:

wSelEnd begins selecting text if not currently selecting, and extends the selection to the end of the current line.

wSelHome()

Comments:

wSelHome begins selecting text if not currently selecting, and extends the selection to the beginning of the current line.

wSelPgUp()

Comments:

wSelPgUp begins selecting text if not currently selecting, and extends the selection up one screenful from the current position.

wSelPgDn()

Comments:

wSelPgDn begins selecting text if not currently selecting, and extends the selection down one screenful from the current position.

wSelWordLeft()

Comments:

wSelWordLeft begins selecting text if not currently selecting, and extends the selection from the current position to the beginning of the next complete word (whitespace delimited text).

wSelWordRight()

Comments:

wSelWordRight begins selecting text if not currently selecting, and extends the selection from the current position to the end of the next complete word (whitespace delimited text).

wSelTop()

Comments:

wSelTop begins selecting text if not currently selecting, and extends the selection up to the beginning of the document.

wSelBottom()

Comments:

wSelBottom begins selecting text if not currently selecting, and extends the selection to the end of the document.

wInvertCase()

Comments:

wInvertCase switches the case of the character at the current position if there is no selection, or of the entire selection if there is a selection.

wUpperCase()

Comments:

wUpperCase changes the case of the character at the current position if there is no selection, or of the entire selection if there is a selection, to all uppercase characters.

wLowerCase()

Comments:

wLowerCase switches the case of the character at the current position if there is no selection, or of the entire selection if there is a selection, to all lower case characters.

wCutAppend()

Comments:

wCutAppend cuts the selected text from the document and appends it to whatever is in the clipboard, rather than replacing what is in the clipboard as wCut does.

wCutMarked()

Comments:

wCutMarked cuts all lines in the current document that are bookmarked and copies the cut text to the clipboard.

wCopyMarked()

Comments:

wCopyMarked copies all lines in the current document that are bookmarked to the clipboard.

wCopyAppend()

Comments:

wCopyAppend copies the selected text from the document and appends it to whatever is in the clipboard, rather than replacing what is in the clipboard as wCopy does.

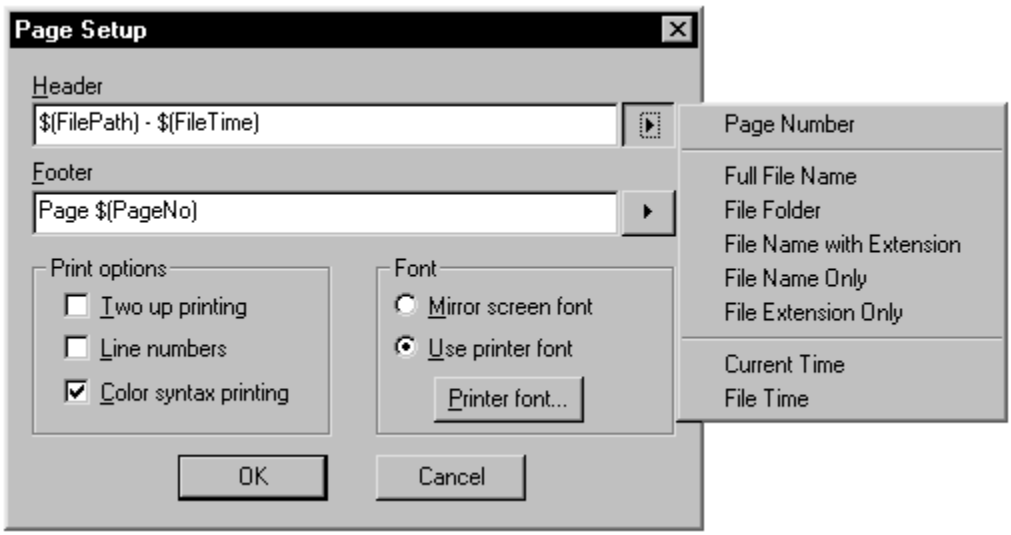
wSetBookmark()

Comments:

Places a bookmark on the current line if one does not already exist, or removes it if it does..

Page Setup

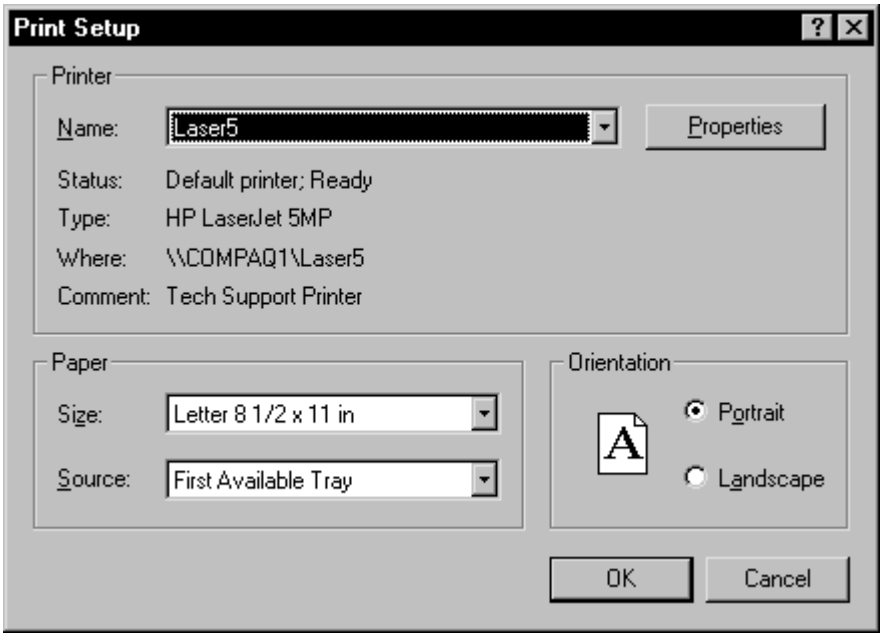
The following options allow you to set print options for each page that will be printed. Click on a dialog box item for more information.



Print Setup

Use this command to select a printer and a printer connection. This command presents a Print Setup dialog box, where you specify the printer and its connection.

Click on a dialog box item for more information or scroll down for a complete list of option descriptions.



Printer Select the printer you want to use. Choose the Default Printer; or choose the Specific Printer option and select one of the current installed printers shown in the box. You install printers and configure ports using the Windows Control Panel.

Orientation Choose Portrait or Landscape.

Paper Size Select the size of paper that the document is to be printed on.

Paper Source Some printers offer multiple trays for different paper sources. Specify the tray here.

Options Displays a dialog box where you can make additional choices about printing, specific to the type of printer you have selected.

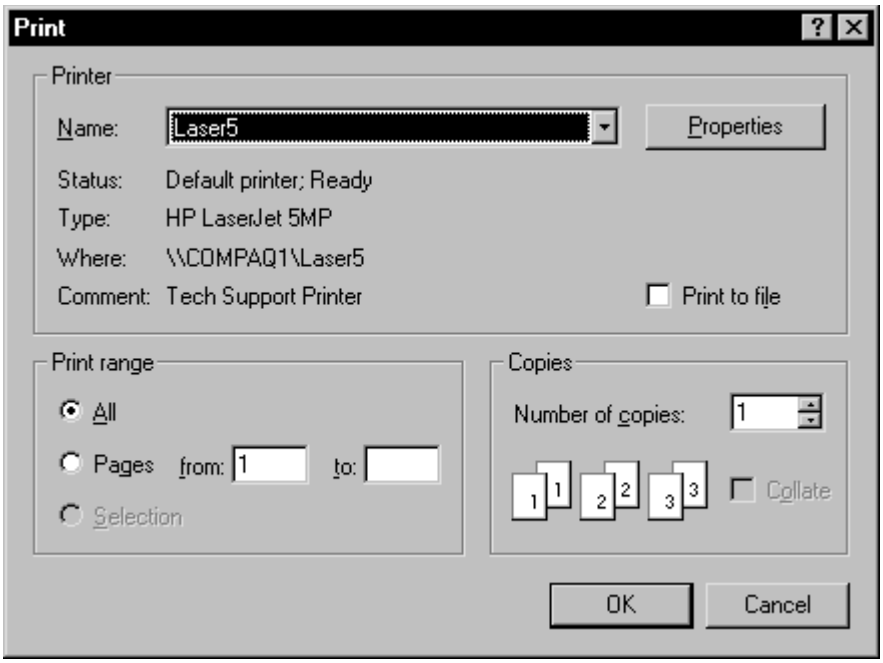
Network Choose this button to connect to a network location, assigning it a new drive letter.

Print

Use this command to print a document.

The Print command presents the Print dialog box, where you may specify the range of pages to be printed, the number of copies, the destination printer, and other printer setup options.

Click on a dialog box item for more information or scroll down for a complete list of option descriptions.



Printer	This is the active printer and printer connection. Choose the Setup option to change the printer and printer connection.	
Properties	Displays a Properties dialog box, so you can select a printer and printer connection.	
Print Range	Specify the pages you want to print.	
	All	Prints the entire document.
	Selection	Prints the currently selected text.
	Pages	Prints the range of pages you specify in the From and To boxes.
Copies	Specify the number of copies you want to print for the above page range.	
Collate Copies	Prints copies in page number order, instead of separated multiple copies of each page.	

Print Progress Dialog

The Printing dialog box is shown during the time that WinEdit is sending output to the printer. The page number indicates the progress of the printing.

To abort printing, choose Cancel.

Print Preview

Use this command to display the active document as it would appear when printed. When you choose this command, the main window will be replaced with a print preview window in which one or two pages will be displayed in their printed format.

The print preview toolbar offers you options to view either one or two pages at a time; move back and forth through the document; zoom in and out of pages; and initiate a print job.

The number of pages displayed in Print Preview also depends upon whether the Two Up printing mode is selected in the Page Setup dialog box. If Two Up is selected, the font size will automatically become smaller and the orientation of the page will change to Landscape.

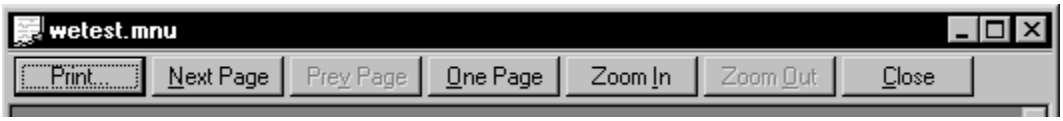
One Up Printing /
One Page Preview

One Up Printing /
Two Page Preview

Two Up Printing /
Two Page Preview

Print Preview Toolbar

The print preview toolbar offers you the following options:



Print	Bring up the print dialog box, to start a print job.
Next Page	Preview the next printed page.
Prev Page	Preview the previous printed page.
One Page / Two Page	Preview one or two printed pages at a time.
Zoom In	Take a closer look at the printed page.
Zoom Out	Take a larger look at the printed page.
Close	Return from print preview to the editing window.

WinEdit Help File

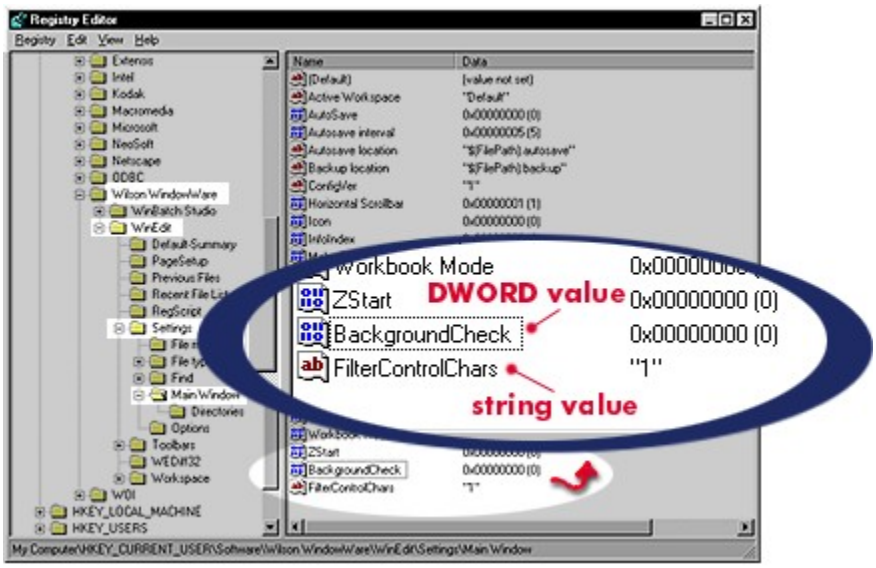
WinEdit Key Mappings

Key	Normal	Shift	Control	Shift +Control	Alt
F1	Help				
F2	Bookmark Next	Bookmark Previous	Bookmark Toggle		
F3	Find Next				
F4	Copy Line to Clip	Previous Error			
F5					
F5	Debug Go				
F7			Project Compile		
F8			Edit Col Block		
F9	Debug Breakpoint		DB: Remove All Breakpoints		
F10	Step Over		DB: Run to Cursor		
F11	Step Into				
F12					
a			Select All		
b					
c			Copy	Copy Append	
d					
e					
f			Search Find		
g			Go to Line		
h			Search Replace		
i					
j					
k					
l			Lower		
m			Match Brace, etc		
n			File New		
o			File Open		
p			Print		
q			Macro Record		
r					
s			File Save		
t					
u			Upper		
v			Paste		
w					
x			Cut	Cut Append	
y			ReDo		
z			UnDo		
Home		Select Home	Top of File		
End		Select End	Bottom of File		
Left		Select Left	Word Left		

Right	Select Right	Word Right
Up	Select Up	
Down	Select Down	
Scrlock		Debug Break

Special Help

Some special configuration items are hidden in the Registry: **Background File Checking** and the **Reading of Null Characters**. See the documentation below for specific instructions.



Background File Change Checking

Winedit checks open documents periodically in the background, to determine if the file on disk has changed since it was opened in WinEdit. In certain situations, such as a slow network or floppy drives, you may wish to prevent this check from occurring.

To turn off WinEdit's background file change checking, add the following DWORD value, **BackgroundCheck**, to the Main Window key in the Registry:

HKEY_CURRENT_USER\Software\WinEdit Software Co.\WinEdit\Settings\MainWindow\ BackgroundCheck

Reading Null Characters

WinEdit 99 does not allow null characters in a file to be read. When null characters are found, the user is prompted to either replace them with the ASCII value 255, or abort the file read.

By adding the value **FilterControlChars=1,** to the Main Window key in the Registry all control characters except tab, carriage return, and line feed will also be filtered in the same manner.

HKEY_CURRENT_USER\Software\ WinEdit Software Co.\WinEdit\Settings\Main Window\FilterControlChars=1

WinEdit Help File

Copyright

Acknowledgments

This software designed by Steve Schauer and Morrie Wilson.

Documentation and Help written by Tina Browning.

Windows Interface Language Help File 

What is WIL?

Windows Interface Language (WIL) is an easy-to-use yet very powerful general-purpose programming language with over **500 functions** for file management, sending keystrokes, disk drive management, directory management, binary file access, multimedia support, DDE support, clipboard handling, system control, program management, string handling, displaying information, user prompting, window management, floating point & integer arithmetic, execution control and more. Many operations that require pages of code in other programming languages can be accomplished with a single WIL function call.

WIL scripts are written in a plain text file, which can be created by Notepad or most word processors. (Of course, we recommend our own WinEdit, which has many features designed expressly for programmers, including a full-featured implementation of WIL itself.)

These text files can take one of two forms, depending on your particular implementation of WIL: **batch files** or **menu files**.

Batch Files

A batch file is simply a list of WIL commands and function calls, executed in order (just like the old DOS batch language).

Menu Files

A menu file is similar to a batch file, except that multiple chunks of WIL code are organized into menu and sub-menus, and each routine is launched by pressing the appropriate keystroke or selecting an item from the menu. (The name and location of the menus vary depending on the particular implementation of WIL menu files.)

Windows Interface Language Help File 

Step by step guide to learning WIL

The Windows Interface Language (WIL) is a scripting language. In order to use it, you must open up an editor and **Create a Script** using the WIL commands. Once written, the script is saved and **run** with an extension already associated with the WIL interpreter. In our examples, we use the extension .WBT.

The WIL language is not hard to learn. A general knowledge of batch file programming is helpful, but not necessary.

Suggestions for Tutorial use

Everyone has different learning styles. The contents of the WIL Tutorial can be accessed in several ways.

Topic by Topic - Arranged so each new concept builds on the last. Scroll through the topics from the top or select the ones which catch your eye.

Step by Step Tutorial Course - For those who have the general idea and don't want to be bogged down with the absolute particulars. Follow along and write a working script.

The Complete Tutorial - For some, the printed word is mightier than the hypertext jump. Here the tutorial has been arranged for easy printing.

The WIL Tutorial

Windows Interface Language Help File 

Menu Files

WIL scripts can be implemented in two ways: via **batch files** or menu files. In a batch process, WIL scripts are associated with the WIL processor, allowing them to be initiated and run on the desktop just as any true executable is launched and run.

WIL scripts can also be launched as menu items from a drop down menu. However, you must have an implementation of WIL with the capability of generating the menu either within one of our applications or as an enhancement to standard Windows applications. In Windows 95/98/NT, WIL adds menu capability to the Windows Task Bar and the Shortcut Menu in the Windows 95/98/NT Explorer.

Please see either the help file or printed documentation that came with your program for more information.

WinEdit Help File

Contacting Winedit Software Co.

Winedit Software Co.
PO Box 1435
Hilo, HI 96721 USA

Orders: (800) 699-6395
Voice: (800) 595-3248
Fax: (808) 934-8314

Email: info@winedit.com…..

Registered users of our software receive: manuals, technical support, use of on-line information services, and special offers on new versions of our products.

How to get technical support

The Winedit website is an excellent technical resource. .You will find additional up-to-date topics on Java setup, adding new file types, and more. You can also check for newer releases of WinEdit. Your WinEdit 99 or WinEdit 2000 registration information will work with all future maintenance releases.

See the information on **registering your copy** if you haven't done so yet.

Internet Web page: http://www.winedit.com

Internet Technical Support Articles:
http://www.winedit.com/support.html
Internet FTP: ftp.winedit.com

Windows Interface Language Reference

Windows Interface Language (WIL) is an easy-to-use yet very powerful general-purpose programming language with over 500 functions for file management, sending keystrokes, disk drive management, directory management, binary file access, multimedia support, DDE support, clipboard handling, system control, program management, string handling, displaying information, user prompting, window management, floating point & integer arithmetic, execution control and more.

WinEdit can also access the Windows Interface Language Help file via its context menu. In your script, highlight a function name then click with the right mouse button. The context menu will open as a drop down list. Select the option "Keyword Help". The help file will launch and display the function page you selected.

Registering your software

Registered users of our software receive: manuals, **technical support**, use of WinEdit Software Co. on-line information services, and special offers on new versions of WinEdit and other products.
You can register online, through our secure commerce server:

http://www.winedit.comYou can register your software by mailing, faxing, or telephoning your registration information to WinEdit Software Co.

Winedit Software Co.
PO Box 1435
Hilo, HI 96721 USA

Orders: (888) 595-3248
Voice: (808) 595-3248
Fax: (808) 934-8314

Ordering Information

Licensing our products brings you wonderful benefits. Some of these are:

- Gets rid of that pesky reminder window that comes up when you start up the software.
- Entitles you to one hour free phone support for 90 days (Your dime).
- Ensures that you have the latest version of the product.
- Encourages the authors of these programs to continue bringing you updated/better versions and new products.
- Gets you on our mailing list so you are occasionally notified of spectacular updates and our other Windows products.
- And, of course, our 90-day money back guarantee.

We have contracted with another company, NorthStar Solutions, to process our orders. The easiest way to order is to use your Discover, Visa, Mastercard, or American Express card and submit your order quickly, conveniently, and securely on the Internet at

http://www.winedit.com

To telephone your order, call WinEdit Software Co. at

1-888-595-3248
1-808-934-8199

To mail your order, print the Order Form, and FAX or mail it to:

WinEdit Software Co.
PO Box 1435
Hilo, HI 96721

To FAX your order, use the following telephone number:
FAX: 1-808-934-8314

Please make any checks or money orders payable to "WinEdit Software Co.". US Currency only, drawn on U.S. Banks, please.

- Order form
- Contacting WinEdit Software Co.

WINEDIT SOFTWARE CO. ORDER FORM

WINEDIT SOFTWARE CO.
PO Box 1435
Hilo, HI 96721

Order Lines:
(888) 934-8199
(808) 934-8199
Fax - (808) 934-8314

- Ordering Information
- Contacting WinEdit Software Co.

Name: _____

Company:_____

Address:_____

_____

City: _____ St:_____ Zip:_____

Phone: (_____)_____
Country:_____

Products

____ WinEdit 2000 – Full Package @ $99.00 : _____.____
____ WinEdit 2000 – Electronic Download @ $89.00 : _____.____
____ WinEdit 2000 – 5 User Pack @ $425.00 : _____.____
____ WinEdit 2000 – 25 User Pack @ $1495.00 : _____.____

Shipping (each copy, except Electronic Download)
____ US and Canada shipping @ $9.00 : _____.____

____ Foreign air shipping (except Canada) @ $20.00 : _____.____

Total : _____.____

Please enclose a check payable to WinEdit Software Co., or you may use Amex, Visa, MasterCharge, or Discover. For credit cards, please enter the information below:

Card #:__ __ __ __ - __ __ __ __ - __ __ __ __ - __ __ __ __ Expiration date: ____/____

Signature: _____

Where did you hear about or get a copy of our products?

_____

International customers please note, payments must be made in U.S. Currency, drawn on a

U.S. Bank.

noyesyesyesWinEdit PopupTRUEWEPOPUPyesyes23/10/98

Table of Contents

Help file produced by **HELLLP!** v2.7 , a product of Guy Software, on 10/23/98 for WILSON WINDOWWARE, INC..
The above table of contents will be automatically completed and will also provide an excellent cross-reference for context strings and topic titles. You may leave it as your main table of contents for your help file, or you may create your own and cause it to be displayed instead by using the I button on the toolbar. This page will not be displayed as a topic. It is given a context string of _._ , but this is not presented for jump selection.
HINT: If you do not wish some of your topics to appear in the table of contents as displayed to your users (you may want them ONLY as PopUps), move the lines with their titles and contexts to below this point. If you do this remember to move the whole line, not part. As an alternative, you may wish to set up your own table of contents, see Help under The Structure of a Help File.
Do not delete any codes in the area above the Table of Contents title, they are used internally by HELLLP!

Page Setup

Header / Footer

Select optional Header/Footer print fields from the drop down list.

Print Options

Two up Printing	Prints two pages side by side in Landscape format. (Font may appear smaller than selected Screen or Printer font.)
Line numbers	Prints line numbers in the left hand margin.
Color syntax printing	Prints document using the designated color syntax highlighting for that file.

Font

Mirror Screen Font	Emulates the Screen font while ignoring Printer Font information.
Use Printer Font	Uses the font selected in the Printer font dialog.

OPTIONS Editor

Restore workspace at startup

Reload all documents that were open at the end of the last editing session.

Leave cursor at start of pasted text

When checked, the cursor (caret) is positioned at the beginning of the pasted text. When unchecked, it is placed at the end of the pasted text.

Allow multiple instances of WinEdit

When unchecked, double clicking an associated file or starting WinEdit itself activates the already running instance instead of launching an additional copy.

Show Horizontal Scrollbar

When unchecked, no horizontal scrollbar is shown.

Allow Virtual Whitespace

When checked, the caret can be positioned in any column. When unchecked, the caret cannot be moved beyond the end of the text of any line

Make backup files

When checked, a backup copy of a document is made whenever the document is saved.

Backup specification

Create a file specification to use when naming a backup file. Choose options based on the original document name from the menu button.

Automatically backup files

When checked, a backup copy of a document is made automatically at the time interval selected.

Autosave file specification

Create a file specification to use when naming an autosave file. Choose options based on the original document name from the menu button.

Options FileType dialog box

File type
Chooses a file type from the dropdown list.

Edit
Allows adding or deleting of file types from the list.

Syntax coloring

Sets options for coloring keywords for this file type, and for specifying the characters which flag text as a ‘comment’.

Font

Selects a fixed pitch font to be used to display this file type.

Tab size

Sets the number of columns each tab character represents.

Show tabs
When selected, tab characters are displayed on screen.

Insert spaces/Keep tabs

Selects whether to insert a tab character, or a corresponding number of space characters, when pressing the tab key.

Insert mode

When checked, text typed is inserted at the caret position. When unchecked, overtype mode is used, where text typed replaces text at the caret position.

Line End

Selects the characters that are inserted when the Enter key is pressed.

Compile command

Enter the command line used to compile this file type. Command line parameters based upon the original document name can be selected from the menu button.

Redirect to output window

Captures command line output and displays it in the output window when this file type is compiled from within WinEdit. Double click on an error or warning or choose Next Error/Previous Error from the Search menumenu_search to move to that position in your source code file.

Custom error parsing

If the compiler output from this file type does not match the format “filename(lineno):text”, you can process the output in a WIL script filehowto_writewil so that Next Error/Previous Error will work correctly. The sample script mserror.wbt demonstrates the required processing.

Print Preview

Print

Bring up the print dialog box, to start a print job.

Next Page

Preview the next printed page.

Prev Page

Preview the previous printed page.

One Page / Two Page

Preview one or two printed pages at a time.

Zoom In

Take a closer look at the printed page.

Zoom Out

Take a larger look at the printed page.

Close

Return from print preview to the editing window.

Print Dialog

Printer

This is the active printer and printer connection. Choose the Setup option to change the printer and printer connection.

Properties

Displays a Print Properties dialog box, so you can select a printer and printer connection.

Print Range

Specify the pages you want to print:

All	Prints the entire document.
Selection	Prints the currently selected text.
Pages	Prints the range of pages you specify in the From and To boxes.

Copies

Specify the number of copies you want to print for the above page range.

Collate Copies

Prints copies in page number order, instead of separated multiple copies of each page.

DEBUG

Go

Begins executing the script commands. Execution will continue to the end of the script or until a breakpoint is encountered.

Step Into

Executes the current line of the script. If the current line is a goto, gosub, or call command, execution stops at the first line of the goto, gosub, or call code.

Step Over

Executes the current line of the script. If the current line is a goto, gosub, or call command, all the code at the goto, gosub, or call location is also executed.

Run To Cursor

Begins executing script commands at the current location and continues to the point in the script where the cursor (caret) is located.

Stop Debugging

Stops execution of the script.

Insert/Remove Breakpoint

Inserts a breakpoint at the current line, or removes it if it already exists. When execution of the script is initiated with the Go or Run To Cursor commands, execution will still stop if a line with a breakpoint is encountered.

Remove All Breakpoints

Removes all defined breakpoints in the current script.

PRINT SETUP

Printer

Select the printer you want to use. Choose the Default Printer; or choose the Specific Printer option and select one of the current installed printers shown in the box. You install printers and configure ports using the Windows Control Panel.

Orientation

Choose Portrait or Landscape.

Paper Size

Select the size of paper that the document is to be printed on.

Paper Source

Some printers offer multiple trays for different paper sources. Specify the tray here.

Properties

Displays a dialog box where you can make additional choices about printing, specific to the type of printer you have selected.

Project Customize Tools

Buttons

Add / Remove	Adds or removes items.
Move up / Move down	Changes the position of an item in the list.

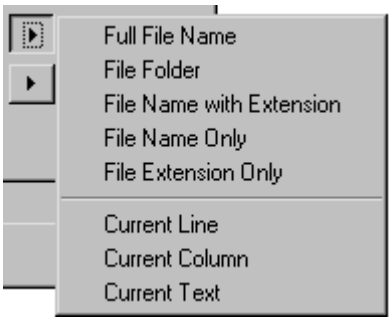
Menu Text

The text which will appear on the Project Menu. An **&** before the first character creates a hotkey.

Arguments

The command line parameters required for the program.

Arguments, can be selected from the drop down list.



Initial Directory

The directory set to be the current directory when the command is executed.

If this screen is the first to display you are probably not using the latest version of the Microsoft HCW help file compiler. You need at least version 4.03 which may be downloaded via a link on http://www.guysoftware.com/helllp.html in order to handle files produced by Microsoft Word version 8 (Office 97) or later.

Unregistered Message

This Help file was produced by an unregistered demonstration copy of the **HELLLP!** file authoring system.

HELLLP! is a user-friendly system to aid in the production of Windows help files. It requires Microsoft Word for Windows version 2.0 or higher. Users of Word versions up to Word 7/Office 95 need version 2.7 of **HELLLP!** users of Word version 8/Office 97 or later need version 3 of **HELLLP!**

HELLLP! is available as shareware from many sources and is always available from http://www.guysoftware.com/helllp.html at which site the author and distributor may also be contacted. Registration and payment, which will remove this screen from files produced by the system may also be accomplished by links from this site.

About Shareware:

Shareware is copyrighted software that is distributed by authors through bulletin boards, on-line services and disk vendors.

Shareware allows you to try the software for a reasonable limited period. If you decide not to continue using it, you throw it away and forget about it. You only pay for it if you continue to use it. Shareware is a distribution method, not a type of software. You benefit because you get to use the software to determine whether it meets your needs, before you pay for it.

The shareware system and the continued availability of quality shareware products depend on your willingness to register and pay for the shareware you use. It's the registration fees you pay which allow authors to support and continue to develop our products. Please show your support for shareware by registering those programs you actually use.