Contents

Overview

Introduction

E! for Windows (EW) is a multi-purpose text editor primarily designed for programmers. However, it will be useful to any PC user who has to deal with text files. E! is not a text processor. It loads and stores ASCII/ANSI files without adding any formatting commands or printer control characters. EW is the Windows version of E!, the full screen DOS editor. EW is not a simple port of E! into the Windows world. It is a fully new product that shares some features with its DOS counterpart.

Although the design of E! is generally CUA compliant, this is not the case for the window management. The MDI interface has many advantages but is rather restrictive when writing a programmer's editor. All text windows are confined within the main application window and this prevents from giving the programmer enough freedom in the way he/she manages the Edit Windows. As you'll see below, E! carries out the idea of a Control Center managing independent Edit Windows. With E! the user has full control over all windows, either individually, as groups or as a whole. This creates a more productive environment.

E! is a simple, ready to use but powerful product. The basic idea behind E! is that most programmers don't want to spend much time programming the tool they use to write programs. Although you can write Extension DLLs to add functions to E! this will not be necessary for a vast majority of users.

E! supports almost any compiler automatically and can be customized to support compilers issuing non standard error messages. E! for DOS users know how easy it is to compile programs from within E!. It's even easier under E! for Windows.

Overview

General Concepts

E! has been designed with the Object-Oriented technology in mind. When you launch E!, a dialog box is displayed. This is actually the application main window. From this dialog box, you will create Editors. Each Editor is responsible for managing one single text. It is an invisible object. This Editor will display at least one Edit Window. It is possible to create Clones, which are secondary Edit Windows allowing to have several views of the same text. Each of these windows has an independent life but can be managed from the Control Center.

In E! there are Global and Local Options. Global Options are shared by all Editors and cannot be adjusted for each Editor individually. Local Options are specific to each Editor and can be changed without affecting the other Editors. Since different texts may require different settings (different margins, different set of macros, a different keyboard, etc...) E! allows you to define a specific environment for each edited file if needed. Otherwise each Editor inherits the Local Defaults defined in the Control Center.

Local Options can be automatically saved for a future session when closing an Editor. This is a convenient way of reloading a file without having to care about Option Settings. The number of sessions over which these data are kept on the disk can be customized (see Modifying EW.INI).

E! is an in-memory only text editor. This provides for faster processing and allows the implementation of very powerful functions but introduces some limitations. In E!, texts cannot be longer than about 16000 lines of 255 characters. We think that this is enough to write programs but this may be a limitation when handling large data files.

Overview

Editor

An Editor is an object created by E! each time you load a file or create a new file. Editors are invisible. They manage a single text image and communicate with the Control Center.

Each Editor creates at least one Edit Window. When all Edit Windows of one Editor are closed, the Editor is destroyed. Hidden Edit Windows are not closed Windows.

See also:

Clones

Clones

Clones are Edit Windows referring to the same Editor. They use the same text image and any modification in one Clone Window is immediately reflected in all other Clone Windows for the same Editor. Clone Windows are used to visualize two or more different parts of the same text.

See also:

Editor
Edit Windows

The Control Center

The Control Center appears whenever you launch E!. This is a dialog box that is divided in two parts.

On the left, the Control Center looks like an ordinary Open Dialog Box. From there you can open files or create new Editors for non existing files. Unlike many Windows applications, E! requires that you give a name for each new Editor that you create. Selecting a filename in the listbox causes information about that file to be displayed immediately.

On the right, a listbox displays a list of all the current Edit Windows. From there you can select a text that you want to edit, temporarily hide or minimize some Edit Windows or organize them on the desktop. You can also save and print files from the Control Center (of course, you can also do that from each Edit Window).

If no Edit Window is currently selected in the listbox, the action that is triggered when clicking on one of the window management buttons applies to all Edit Windows in the listbox. The Clear button clears the current selection. Other buttons have an obvious meaning.

The Control Center Menu
The Control Center Dialogs
Opening Files from the Control Center

See also:

Edit Windows

The Control Center Menu

Most items in the Control Center Menu open dialog boxes. Only the File Menu and the Help Menu have additional options.

File Menu
Help Menu

See also:

Control Center Dialogs
Opening a File from the Control Center

The Control Center File Menu

Set...

This menu opens the File Options dialog box.

Directories...

Search Directories

List here all the directories where E! should look for a file if it doesn't find it in the current directory. Directory names may be separated by commas, blanks or semicolons.

User Directory

E! needs a directory to store all the files it can create during a session. These are not temporary files but keyboard files, macro files or other data files. Keyboards, Keyboard Macros, User Menus, Syntax Expansion Sets, User Extension DLLs and other important data files can only be loaded from that directory. E! automatically defaults to the USER directory created by the installation program.

You may change the USER directory during an editing session if you want to manage different sets of data files.

Save and Load Options

Save Options as...

This option allows you to store the current value of all Global Options to the specified file. You can then reload these options at any time.

Load Options...

Load a Global Option file created using Save Options As. For the sake of consistency, these files should have a .INI extension.

Printer Setup...

Allows you to select a printer and to modify its settings.

Page Setup...

Allows you to define the page layout, the font and the header and footer printed at the top and bottom of each page. The syntax of the header and footer command lines are described in the Printer section.

Grep...

GREP is a common utility that is able to search which files on your disk contain a given character string. E! has a built-in GREP allowing to search and edit the files in the same process. Found files are opened at the exact location in the text where the

first occurrence of the string was found. The E! built-in GREP also supports boolean (regular expression) search.

Create File List...

E! will prompt you for a filename, proposing a "@" as first character. If you click the Ok button, E! will generate a file containing the name of all the files currently edited.

If the "@xxxx.xxx is a List" option has been checked in the File Options dialog box, whenever you'll try to load this file, E! will load the files contained in the list. To edit the list, uncheck the above mentioned option before loading the file.

Recent List

E! maintains a list of the last five files recently edited. This list is appended to the end of the Control Center File Menu and to the end of the Edit Window File Menu. Click on the filename of your choice to edit it.

Control Center

The Control Center Help Menu

Register...

This option allows you to enter your registration code, preventing the reminder box to appear regularly while editing. CRC checking can only be disabled on a registered copy of E!.

UnRegister

This option removes your registration code from EW.EXE allowing you to distribute an unregistered copy to a friend or to register with a new code.

Help Features

Control Center

The Control Center manages the Global Options. The Global Options are set for all the Editors. The default values of the Local Options (which can be set separately from each Editor) are also managed from the Control Center.

The Control Center Dialogs

File Options Dialog
Edit Options Dialog
Search Options Dialog
Local Defaults Dialog
Display Options Dialog
Font Dialog

See also:

Control Center Menu
Opening a File from the Control Center

The File Options Dialog

Log Session

This option indicates that E! should "remember" which files were edited when the Control Center was closed. E! also stores the current Local Options for each file, the Edit Windows position, the Control Center position and the state of each window (minimized, hidden or normal). The next time you'll run E!, everything will be restored in the same state it was when you leaved it.

Bak Files

Check this option if you want E! to create .BAK files each time you save a file.

OverWrite Warning

Check this option if you want a warning when E! is about to overwrite an existing file.

@xxxx.xxx is a List

If this option is activated, E! will not try to load any file which name begins with an ampersand character (@). It will consider that this file is a list and will rather try to load all the files which names are contai ned in the list. This is very convenient when you always work on the same set of files. Please enter one file specification per line. Wildcards are allowed.

Delete Output File

When running a compiler E! will normally display and analyze the compiler output to check for errors and it will delete the output file itself (created by E! by redirecting the compiler output while it is displayed in the DOS window). Check this option to keep the output file (EWX.OUT) on your disk.

Autosave Options

Check this option if you want any change to the Global or Local options to be automatically recorded without having to save the option set manually.

Autosave Files

If this option is activated, E! saves your work in temporary files at regular intervals. This process occurs in the background for all files but the one currently edited. When the current file is autosaved, you have to stop working a few seconds. Files are saved in the TEMP directory if it exists. Otherwise they are stored in the Windows directory (this is the standard Windows behavior for temporary files).

A date and time stamp as well as the "real" filename are always added to the beginning of each file to make their recovery easier. Temporary filenames generated by E! are of the following form: ~EW????.TMP. Other temporary filenames generated by E! (~SPI????.TMP) are used by the Undo system. These files are deleted if E! exits normally.

Delay

This is the interval, in seconds, between autosave operations.

Default Extensions

When you load a file by giving its name directly, you can omit its extension if this extension is mentioned in the Default Extension list. Extensions may be separated by commas, blanks, dots or semicolons. Of course, if a file exists with that name (without extension) it will be loaded first.

Provided your Default Extension List contains the following information: "c; h", If you ask E! to load MYPROGRAM, E! will first load MYPROGRAM (your makefile ?) if it exists. If you ask again to load MYPROGRAM, E! will load MYPROGRAM.C. A subsequent request will cause MYPROGRAM.H to be loaded.

Warning: this feature will only work if the file is located in the current directory or in a directory mentioned in the Search Directories.

See also:

Edit Options Dialog
Search Options Dialog
Local Defaults Dialog
Display Options Dialog
Font Dialog

The Edit Options Dialog

Undo Levels

Any editing function in E! (including cursor movement) is undoable (Default assignment: **Alt+Bsp** or **Ctrl+Z**). An Undo Stack is maintained for each Editor. This consumes memory and hard disk space. E! defaults to 128 Undo Levels. This is enough for most operations.

Be aware that some editing functions call other functions. Undoing these functions always consumes more than one Undo level. So, setting Undo Levels to 1 is not a good idea. A minimum of 5 will make sure thatany function can be submitted to Undo. Set Undo Levels to 0 to disable the Undo System for the next session.

Add on last Line

Check this option if you want E! to automatically add new lines at the end of the text when hitting the Cursor Down key. Otherwise only the Enter key will add new lines.

Enter splits Line

E! has a non-standard behavior when the Enter key is pressed. A new line is inserted leaving the current line unchanged. If you want E! to behave more classically, thus splitting the line at the cursor position when the Enter key is used (and if Insert Mode is On), check this option.

Insert by Default

When checked, this option causes each Editor to be opened in Insert Mode. Otherwise, they are opened in Replace Mode. Please notice that the cursor looks differently while in Replace Mode.

Syntax Expansion

Specify whether the Syntax Expansion feature is enabled. Please see the section relative to Syntax Expansion.

Replace Selection

If text has been selected, pasting text from the clipboard or hitting a character key will normally replace the selected text. You may decide that E! will behave differently by unchecking this option.

AutoScrolling

When AutoScrolling is On, adding a new line in the text will cause the latter to scroll upwards, thus leaving the new line at the same height within the Edit Window. This makes entering text much more comfortable. This occurs only if the cursor is located in the bottom half of the Edit Window.

Beep On Error

Uncheck this one if you don't want to be bothered by a beep whenever an error occurs.

Format From Current Line

The Format function formats the whole paragraph into which the cursor is located. You may specify that you want to format from the current line only by checking this option.

Auto Macros

E! has the ability to automatically execute macros against files having a specific extension if you have created a macro which name **is** that extension. For example, if you have created a macro named PAS (a .EWM extension is mandatory and automatically added), E! will execute that macro each time you load a .PAS file. This feature is not activated by default.

See also:

File Options Dialog
Search Options Dialog
Local Defaults Dialog
Display Options Dialog
Font Dialog

The Search Options Dialog

Global Search

If this option is activated, E! will always search from the beginning of the current text.

Case Sensitive Search

Specify whether the search is case sensitive.

Keep Case

If this option is checked and if the search is case insensitive, the string replacement process will work as follows: each character in the replacement string will be translated to the case type (upper or lower) of the character at the same position in the replaced string.

For example, if the replacement string is "stringb", "StringA" would be replaced with "StringB", "Stringa", would be replaced with "Stringb", "stringA" with "stringB" and so on...

Backward Search

Search from the current cursor location towards the beginning of the text.

Boolean Search

Specify that the search string has to be interpreted as a regular expression.

Block Search Only

Search only within the selected text. Useful for replacing text in a limited area.

Word Only Search

Only occurrences of the search string appearing as a whole word will be found.

Search across Files

E! will search the string not only in the current text but within all active Editors. The search will end when the current location within the active text will be reached again.

Unprompted Replace

During a Search and Replace Operation, E! will replace the found string without prompting the user.

See also:

File Options Dialog
Edit Options Dialog
Local Defaults Dialog
Display Options Dialog

Font Dialog

Standard Search Functions
Boolean Search
Sample Regular Expressions
Boolean Search and Replace
E! Specifics

The Local Defaults Dialog

Local Options are specific to one Editor. We have seen in the previous sections that you can Load and Save Local Options .INI files at will. You should also be aware that E!, when loading a file, will try to find a .INI file in the User Directory with the same name as the filename extension of the file currently loading. If found, E! will read the Local Options from that file.

Strip Trailing Spaces

If this option is On, E! will automatically strip trailing blanks when loading, saving and editing a file.

AutoBackspace

If this option is On, E! deletes the left character when hitting the Backspace key and moves all the text to the right of the cursor one character left. This is the default behavior. Otherwise, if E! is in Replace Mode, the left character is merely replaced by a blank.

AutoInsert

Check this option if you want E! to insert a right brace for you automatically when entering a left brace ('(', '{', or '['). This occurs in Insert Mode only.

AutoTab

If this option is On, E! generates tab positions based on the text of the previous lines. This is the most convenient mode when writing programs. Otherwise, tab positions are based on the values you have entered in the relevant field of this dialog box.

AutoCompress

Within E!, Tab characters have no existence. Every one of them is translated to spaces to allow more powerful block manipulation functions to be implemented. So, when loading a file, E! expands Tab characters to spaces and compressed them back to Tabs when saving the file. This behavior can be disabled using this checkbox. In that case, Tab characters will be considered as "normal" characters and not expanded.

WordWrap

This causes E! to wrap to the next line automatically when the entered text goes beyond the right margin. The next line will begin at the left margin or at the paragraph margin if a new paragraph is created. However, E! doesn't automatically reformat the text if you insert new text in an already formatted paragraph or if you insert text past the right margin.

Use the Format function or the ribbon format button to reformat the paragraph. When WordWrap occurs, E! will position the beginning of the next line at the same margin as the previous line if this line doesn't begin at the paragraph margin or at the left margin. This allows you to easily enter paragraphs with negative indenting without

changing the current margins.

Right Justify

This adds Right Justification to Wordwrap. This option cannot be activated if WordWrap is Off.

Load/Save Ascii

Please see the discussion about ANSI/OEM character sets in the ANSI/OEM section. This topic is important mainly for non english speaking countries.

EOF Mark

If you want that an End Of File (Ctrl Z) mark be added to the end of any file when saving it to disk, check this option. Existing EOF marks are not maintained if this option is Off.

Tabs

Enter here the value of the tab positions that should be used when AutoTab is Off. If these positions are separated by the same intervals only enter the first values. E! will add subsequent values automatically.

For example, if you want the following tab positions to be used: 1 5 9 13 17 etc... just enter 1 5. E! will always extend the tab position set using the interval between the last two values entered in that field. The following set: 1 3 7 9 12 20 will extend to 1 3 7 9 12 20 28 36 44 etc...

Tab Increment

This value, that defaults to 8, is used by E! when expanding or compressing Tab characters. Please don't change it unless you have to load files created using a non standard tabulation increment.

Margins

Enter here three values representing the left margin, the right margin and the paragraph margin respectively.

EOL Sequence

Normally, E! recognizes lines ended with a CRLF sequence (standard). You may want to load some files (i.e. Unix files) that use a different EOL sequence. Select the one suited for the files you are editing.

See also:

File Options Dialog
Edit Options Dialog
Search Options Dialog
Display Options Dialog
Font Dialog

The Display Options Dialog

File shown after loading

This is the default but you may decide to have the editor created and to have it listed in the Control Center Select listbox only. In that case the Edit Window will be hidden. This is useful when loading several files.

Wildcards files shown

Since you can ask E! to load for example "*.PAS", you may not want to have all the Edit Windows appear on your desktop. Uncheck this option to have these windows hidden when loading.

Reshow CC after loading

This option causes E! to redisplay the Control Center after loading a file.

Minimize CC after processing

This option causes E! to minimize the Control Center after any action triggered from it.

Display Shortcuts in Menus

If this option is On, E! displays shortcuts in menus. This is done dynamically to take the current keyboard assignments into account. Any change that you make to the current keyboard of the current Editor is reflected in the Edit Window Menu. This slows down menu display operation a little on low-end systems.

See also:

File Options Dialog
Edit Options Dialog
Search Options Dialog
Local Defaults Dialog
Font Dialog

The Font Dialog

For displaying text in Edit Windows, E! will use fixed-pitched fonts only. That is, non proportional fonts. Proportional fonts are not suitable for a text editor.

Thus, you can only select a fixed-pitched font. E! shows you if the font is an ANSI (A), an OEM (O), a TrueType (T), a Device (D) or a Raster (R) font. It also shows you the different sizes available for this font.

However, even if the text is displayed using non-proportional fonts, you may choose to print it using proportional fonts. That's why the Printer Font Setup dialog has a Fixed-Pitch only checkbox which will allow you to add proportional fonts to the standard list of fonts used by E!.

Fonts with the same size may have different characteristics. E! will show you whether a font is italic, underlined or struck out. Two fonts showing exactly the same description in a font listbox may have a different "weight" (bold, extra bold, etc...). This information is also displayed in the dialog box (as a decimal number) and E! will be able to remember which one you used and will reuse the correct font in a next session (provided the Log Session option is activated).

For TrueType fonts, E! allows you to change the font default size. If a TrueType font is selected, the Font Size edit field will be enabled and you will be allowed to enter a new size. E! will reuse this size when reloading that font.

If you select a new font with the Apply or the Ok button, all Edit Windows are repainted using that font. Use the Apply button to try different fonts or font sizes before closing the dialog box. Otherwise use the Ok button.

See also:

File Options Dialog
Edit Options Dialog
Search Options Dialog
Local Defaults Dialog
Show Options Dialog

Opening Files from the Control Center

There are several ways of opening a file from the Control Center.

The more classical way is to double-click a filename or to click on the Open button while a filename is selected. This is consistent with most Windows programs.

You can also click on the OK button or press enter while a valid filename or filespec is present in the filename field. If the filespec contains wildcards E! will update the listbox accordingly.

If the filename has no extension, E! will try to load a file with the same name and with an extension belonging to the list of default extensions defined in the Default Extension field of the File Options dialog box.

If you click on the New button, E! will prompt you for a name for the new file. If that file exists, it will be loaded. If the name that you provide contains wildcards, E! will try to load all matching files.

Moreover, you may enter several filespecs in the New dialog box. These filespecs must be delimited with spaces, commas or semicolons.

See also:

Control Center Menu
Control Center Dialogs
Opening Files from an Edit Window

The Edit Window

See also:

Control Center Menu

Edit Window Overview

If you open a file, at least one Edit Window is created. An Edit Window has a menu from which you can perform usual editing functions and much more. The Edit Window Menu will be detailed later. E! has a great number of functions. So, each and any one of them could not be inserted in the Edit Window Menu.

However, E! offers you a User Menu that you can customize in many ways. Functions that are missing from the Standard Menu and for which you don't want to learn a shortcut can be added to the User Menu.

Each Edit Window has its own life and cannot be considered as a child of the Control Center. If you minimize an Edit Window, the icon behaves like the icon of a separated application. Of course, when you decide to close the Control Center, all Edit Windows are also closed.

An Edit Window can display an optional ribbon that gives access to commonly used functions through a set of bitmap buttons. Not everyone is convinced that such "ribbons" or "speedbars" are really useful, so you can decide not to display it. This will save System Resources.

The bottom of the Edit Window has four "Status Windows". They display the current cursor position, the total number of lines in the text and the error messages.

See also:

Edit Window Menu

Edit Window Menu

Edit Window File Menu

Open...

Use this dialog box to open a text file. If this file doesn't exist either in the current directory or in the directories listed in the E! Search Directories, E! will ask whether you want to create a new file with that name.

New...

Use this dialog box to create a new file. If the file exists, it will be loaded. Wildcards are allowed as well as filelists. That is, you may enter several file specifications separated by commas, blanks or semicolons. For the sake of security, an Edit Window is never untitled. You must provide a filename before opening an Editor.

Remember the default extension feature: if the files you want to load have an extension that is inserted in the default extensions list, you don't have to specify the extension.

This feature also works from the New button of the Control Center.

Clone

This command creates a new Edit Window for the current Editor. This allows you to see different part of the same file simultaneously. Changes in one Edit Window are immediately reflected in all other Clone Windows.

Get File...

This command allows you to insert a text file in the current text at the cursor position. See also Drag and Drop support.

Write Selection...

If a selection is active you can write it to the file of your choice.

Append Selection...

Same as Write Selection but the selected text is appended at the end of the specified file.

Save

Save the file under its current name.

Save as...

Save the file under the specified name. **The current name is not changed**.

This behavior differs from other Windows applications. Since many active data are related to the filename and particularly to the filename extension, changing the filename when using Save As... would result in confusing operations regarding the User Sets in general.

Refresh

This command reloads a fresh copy of the file. You will be asked whether you want the original file or the most recent temporary file created by the Autosave feature to be reloaded.

Grep...

Same function as in the Control Center.

See the Grep section.

Associate...

This dialog box allows you specify which keyboard, user menu, syntax expansion set and compiler command set will be automatically associated with the files having the extension specified in the Filename Extension field. Change the default (current) filename extension if necessary and enter the name of the User Sets you want to be associated.

See Associations for more details.

Switch to...

E! maintains a list of all the current Edit Windows. This allows you to switch from one text to another without having to use the Control Center.

Control Center...

Display the Control Center.

Print

Print the file according to the current printer settings.

Printer Setup...

Allows you to select a printer and to modify its settings.

Page Setup...

Allows you to define the page layout, the font and the header and footer printed at the top and bottom of each page. The syntax of the header and footer command lines are described in the Printing section.

Close Editor

Close All Edit Windows for that Editor and destroy the Editor.

Exit E!

Verify if all modified texts have been saved and exit.

Recent List

Same function as in the Control Center.

See also:

Edit Menu
Bookmark Menu
Search Menu
Compile Menu
Local Menu
Windows Menu
User Menu

Edit Window Edit Menu

Undo

Undo the last cursor movement or the last text modification. You can undo as many actions as you have set in the Control Center Edit Options dialog box. Remember that some actions use several Undo Levels. Default assignment: **Alt+Bsp** or **Ctrl+Z**.

Clear Undo Stack

This option causes the Undo System to be reset. All previous changes will not be undoable any more. Use this option if you want to begin a series of changes that are very likely candidate to be undone as a whole.

Cut

Insert the current selected text into the clipboard. Any existing data in the clipboard will be lost. The selection is deleted from the current text. Default assignment: **Shift+Del** or **Ctrl+X**.

Cut & Append

Append the selected text to the clipboard if it already contains text. The selection is deleted from the current text.

Copy

Works like Cut but the selection is not deleted. Default assignment: **Ctrl+Ins** or **Ctrl+C**.

Copy & Append

Works like Cut & Append but the selection is not deleted.

Paste & Insert

Insert the text contained in the clipboard at the current cursor position. Default assignment: **Shift+Ins** or **Ctrl+V**.

Paste & Overlay

Overlay the text at the current cursor position with the text contained in the clipboard. This feature works only in Line and Block selection mode.

Clear

Clear the current selection.

Select All

Select the whole text in Line Mode.

Delete

Delete the current selection or the character at cursor. Default assignment: **Del**.

Draw...

This command will first display the Draw Mode dialog box and then will put E! in the selected Draw mode.

Begin / End Macro

Begin / End keyboard macro recording. Default assignment: **Ctrl+M**.

Load Macro...

Load a macro from the E! directory.

Save Macro...

Save a macro to the E! directory. You just need to give an 8 characters name without any extension.

Replay Macro

Replay the current keyboard macro. Default assignment: **Ctrl+Y**.

Ascii Mode

Ctrl+character keystrokes normally trigger one of the built-in E! function. If you want to insert the actual control character in your text, just activate the Ascii Mode.

See also:

File Menu
Bookmark Menu
Search Menu
Compile Menu
Local Menu
Windows Menu
User Menu

Edit Window Bookmark Menu

Please see the Bookmark section for an overview of bookmarks.

Set

Create a bookmark at the current cursor position. You can create up to 3 bookmarks per Edit Window.

Goto Bookmark

Goto to the specified bookmark.

Remove Bookmark

Remove the specified bookmark. Bookmarks are easier to use from the ribbon. Create a bookmark by clicking on the Exclaim mark button. Then go to the bookmark by clicking on the relevant bookmark button (green mark). Clicking on a bookmark button while the cursor is at the bookmark position will remove the bookmark.

Goto Line...

Go to the specified line. Default assignment: **Ctrl+G**.

Goto Selection

Go to the first character of the active selection. Default assignment: **Ctrl+O**.

Goto End of Selection

Go to the end of the active selection. If the selection mode is "Line Selection", the cursor is moved to the beginning of the last selected line. Default assignment: **Ctrl+Shift+O**.

Compare

Please see the Comparing two files section. Comparison is always triggered from the Control Center.

See also:

File Menu
Edit Menu
Search Menu
Compile Menu
Local Menu
Windows Menu
User Menu

Edit Window Search Menu

Please see the Searching section for an overview of Search functions in E!.

Find...

This command opens the Search dialog box. Default assignment: **F5**.

Repeat Last

Repeat last search. Default assignment: **Shift+F5**.

Find Word

Find the next occurrence of the word at the current cursor position. Default assignment: **F6**.

See also:

File Menu
Edit Menu
Bookmark Menu
Compile Menu
Local Menu
Windows Menu
User Menu

Edit Window Compile Menu

Please see the Compiling section for an overview of Compilation support within E!.

Compile

Run the Compile command as defined in the Commands dialog box.

Make

Run the Make command as defined in the Commands dialog box.

Rebuild

Run the Rebuild command as defined in the Commands dialog box.

Debug

Run the Debug command as defined in the Commands dialog box.

Run

Run the Run command as defined in the Commands dialog box.

Next/Previous Error

Go to the next/previous error detected by E! after a compilation process.

Commands...

This dialog box allows you to define which compilers will be used against files having a specified extension. Please see the Compiling section to learn how E! supports compilers.

See also:

File Menu
Edit Menu
Bookmark Menu
Search Menu
Local Menu
Windows Menu
User Menu

Edit Window Local Menu

Set...

This command opens the Local Defaults dialog box. But this time, changes will apply to the current Editor only. Local Defaults defined in the Control Center will continue to be applied to any new Editor.

Save Local Options as...

This command allows you to save the Local Options of the current Editor into a .INI file for later use.

Load Local Options...

Reload Local Options from a file created with Save Local Options As. These Options will apply to the current Editor only.

Key Assignment...

This command opens the Key Assignment dialog box. This is the place where you can modify your keyboard layout by assigning Functions, Keyboard Macros or Extension DLLs to a key. E! Keyboards can be saved and loaded at a ny time. Each Editor can have its own keyboard. Please see the Keyboard Assignment section.

Syntax Expansion...

This command opens the Syntax Expansion dialog box. Syntax Expansion allows you to bind a keyboard macro to any character string. For example, you can decide that "if" followed by a blank will trigger a keyboard macro entering the whole "else" structure automatically. Please see the Syntax Expansion section.

See also:

File Menu
Edit Menu
Bookmark Menu
Search Menu
Compile Menu
Windows Menu
User Menu

Edit Window Windows Menu

Ribbon

Click on that menu item to display or hide the toolbar (ribbon).

Hide Window

Hide the current Edit Window. It can be redisplayed from the Control Center.

Tile

Tile the Edit Windows either by rows, columns or both.

Cascade

Cascade the Edit Windows.

Hide All

Hide all Edit Windows for that Editor. To hide all Edit Windows for all Editors, use the Control Center.

Minimize All

Minimize all Edit Windows for that Editor. To minimize all Edit Windows for all Editors, use the Control Center.

Synchronous Scrolling

Toggle this option to have all Edit Windows scrolling simultaneously when you use the scroll bars or a scrolling function (unless they are minimized).

Font...

Open the Font Selection listbox.

Next Editor...

Trigger a jump to the next Editor available (except Editors for which the first Edit Window is minimized or hidden. If you assign this command to a key, this will allow you to quickly switch from one text to another without using the switch list (provided this key assignment is common to all the keyboards you are using).

See also:

File Menu
Edit Menu
Bookmark Menu
Search Menu
Compile Menu
Local Menu
User Menu

Edit Window User Menu

Load User Extension...

Load a user-written Extension DLL.

Unload User Extension...

Unload a previously loaded Extension DLL.

Execute User Extension...

Allows execution of a user extension if it has not been assigned to a key.

Autoload...

Define the list of User-written Extension DLLs that will be automatically loaded upon startup.

Since these DLLs are stored in the USER directory you only have to give an eight character name. However, a full pathname is allowed.

Define User Menu...

Define the user commands that will be added to the user menu. These commands can trigger execution of an Editing Function, a Keyboard Macro, a Windows or a DOS program or a User-written Extension DLL. Menus can be saved and restored at any time. Please see the User Menu section.

See also:

File Menu
Edit Menu
Bookmark Menu
Search Menu
Compile Menu
Local Menu
Windows Menu

Each Edit Window inherits the default properties and settings of the other Edit Windows for that Editor or the default properties and options defined in the Control Center. These default options can be modified using the Local Options dialog box.

Edit Window Dialogs

- Grep dialog
- Page Setup dialog
- Draw Mode dialog
- Search dialog
- Compiling Commands dialog
- Local Options
- Keyboard Assignment Dialog
- Syntax Expansion dialog
- User Menu dialog

Edit Window Ribbon

Some editing functions and commands can be triggered by clicking on one of the bitmap buttons of the Edit Window Ribbon. Following is the description of these buttons.

Save the current file. Appears in green if the file has been modified.

Copy the current selection to the clipboard. Appears in green if text has been selected.

Paste the clipboard text at the cursor position. Appears in green if the clipboard contains text.


Find command


Repeat Find command


Find Word command

Activate the WordWrap mode.

Activate the Justification mode.

Reformat the current paragraph.

Display the Draw Mode dialog box or deactivate Draw Mode.

Translate the current selection or the current word to Uppercase.

Translate the current selection or the current word to Lowercase.

Activate the Fill Block function.

Activate the Shift Block function.

Activate the Sort function.

Activate the Add Block function.

Create a bookmark at cursor position.

Goto to an existing bookmark.

If you have forgotten the meaning of a ribbon button, you can click on it with the right mouse button. A Help message will be displayed.

Opening a File from an Edit Window

The Open option from the Edit Window File Menu allows you to open file by specifying their name. Like from the Control Center, Default Extensions are taken into account.

Likewise, the New option allows you to create new files or to load existing files. You are also allowed to specify filenames containing wildcards and to omit the file extension if it belongs to the Default Extension list.

Moreover, you may enter several filespecs in the New dialog box. These filespecs must be delimited with spaces, commas or semicolons.

See also:

Opening Files from the Control Center

Scrolling and Moving the Cursor

Using the Keyboard

Direction Keys

These keys move the cursor one character left or right or one line up or down. The text window scrolls accordingly if necessary.

Ctrl Left / Right

These keys move the cursor to the previous or to the next word in the current text. Words are not delimited with blanks only. E! word parsing is context dependent and you'll see that, depending on the user request, E! will sometimes parse the line somewhat differently.

Home / End

Move the cursor to the beginning or to the end of the current line respectively. If the cursor is already located at the beginning or at the end of the line it will be moved to the beginning of the previous line or to the end of the next line.

Ctrl+Home / End

Move the cursor to the beginning or to the end of the current text. If you press **Ctrl+Home** while the cursor is already on the text's first line, it will be moved to the beginning of the line. If you press **Ctrl+End** while the cursor is already on the text's last line, it will be moved to the end of that line.

Ctrl+PageUp / PageDown

These keys move the cursor to the top or to the bottom of the current window respectively. The column position is not changed.

Ctrl+O

Go to the beginning of the active selection.

Ctrl+Shift+O

Go to the end of the active selection. If the selection mode is "Line Selection", the cursor is moved to the beginning of the last selected line.

Ctrl+P / Ctrl+Shift+P

Move the cursor to the next or previous paragraph respectively. Paragraphs are delimited by empty lines.

Tab / Shift+Tab

If AutoTab is Off, move the cursor to the next / previous Tab position defined in the Local Options dialog box. Otherwise, tab positions are based on the beginning of words in the previous lines. E! automatically finds the most logical position based on the lines of text preceding the current line.

PageUp / PageDown

Scroll the text window one screen up or down.

Alt+Shift+Direction Keys

Scroll the text window one character left or right or one line up or down.

F9 / F10

Scroll the text window one screen left or right.

Shift+F8

Put the current line at the top of the text window.

Shift+F9

Put the current line in the middle of the text window.

Shift+F10

Put the current line at the bottom of the text window.

Please be aware that the cursor will always "follow" the text window. Unlike other programs (like Windows Write) that don't change the cursor position even after scrolling up or down, E! always computes a new cursor position if scrolling occurs. Since E! has a bookmark feature, it is not necessary to leave the cursor position unchanged while scrolling.

See also:

Using the Mouse
Bookmarks

Using the Mouse

The text window scrollbars work as expected in any Windows program. E! gives you a little more however regarding scrolling with the mouse.

Clicking on the mouse left button will put the cursor at the current mouse cursor location. However, this doesn't happen the first time you click with the left button in the text area. This allows you to give the focus to the Edit Window without moving the cursor. This will be useful especially when giving the focus to Edit Windows opened after a Grep operation because Grep positions the cursor at the location where the search string has been found in the text. The right mouse button behaves normally. So you have the choice between both solutions.

Clicking on the mouse left button and dragging in any direction will begin text selection in Stream Mode and will allow the text window to scroll if necessary. The greater the distance between the mouse cursor and the window border, the faster the text window will scroll.

As usual, clicking on the left mouse button will clear the current selection. If you want to change the cursor position without clearing the current selection, use the right mouse button.

Accordingly, clicking and dragging with the right mouse button will allow mouse scrolling without beginning a text selection or without clearing the existing selection.

Changes to the cursor position made using the mouse are also undoable.

See also:

Using the Keyboard
Bookmarks

Bookmarks

Bookmarks are used to memorize cursor positions when you need to move quickly between different parts of the text.

Bookmarks are created either by using the Set option of the BookmarkMenu or by clicking on the Bookmark Button in the ribbon. The bookmark is always created at the current cursor position.

You may create up to 3 bookmarks per Edit Window. To go back to a bookmark you can either select the Goto Bookmark option from the Bookmark menu or merely click on the green Bookmark button created in the ribbon. If you click on that button while the cursor is already located at the bookmark position, the bookmark is destroyed.

See also:

Using the Keyboard
Using the Mouse

Editing Functions

Not all E! editing functions can appear in the Edit Window Menu. This would be too much. Moreover, all these functions are not always needed. So it would be useless to have them permanently inserted in the Edit Window Menu. Here is the description of all functions that have not been described above. Each of these functions can be assigned to a keystroke or inserted in the User Menu.

Insert Macro Text

This function allows to insert either the current system date, system time or filename at the cursor position. Default assignment: **Alt+Shift+D**, **Alt+Shift+T** and **Alt+Shift+F** respectively.

DeleteLeft

Delete the character at the left of the cursor. If the insert mode is off and if the AutoBackspace function is off, the text at the right of the cursor is not moved and the deleted character is replaced with a space. Default assignment: **Backspace**.

Shift Text

Shift the text at the right of the cursor one character to the right without moving the cursor. Default assignment: **Ctrl+=**.

Align

Move the first non blank character of the line to the cursor position. Default assignment: **Ctrl+A**.

Insert Brace

Try to balance the braces ('<>', '{}', '()', '[]') existing on the current line by inserting the missing right or left brace. This function works in Insert mode only if the AutoInsert mode is active. If there is an ambiguity in resolving brace insertion, nothing happens. Default assignment: **Ctrl+B**.

ClearLine

Clear the contents of the current line. Default assignment: **Ctrl+N**.

DeleteLine

Delete the current line. Default assignement: Ctrl+D.

Delete End Of Line

Delete the end of line from the cursor position. Default assignment: **Ctrl+E**.

Upcase First

The current word is changed as follows: the first character and all characters following an underscore or dash character are translated to uppercase while all other characters are translated to lowercase. This function is mainly intended for C and

Pascal programmers. Default assignment: **Ctrl+F**.

Uppercase

If a selection is active all the selected text is translated to uppercase. Otherwise the function applies to the word at the cursor position. Default assignment: **Ctrl+F1**.

Lowercase

If a selection is active all the selected text is translated to lowercase. Otherwise the function applies to the word at the cursor position. Default assignment: **Ctrl+F2**.

Reflow

If the current Editor is in WordWrap mode (or Justify mode), all the paragraphs included in the current selection are formatted according to the current margins. If no selection is active only the current paragraph is formatted.

If the selection is a Block, the current margins are temporarily replaced by the margins defined by the block thus allowing a different style for that part of the text. If the Format From option has been activated, the current paragraph is formatted only from the current line. Default assignment: **Ctrl+F7**.

Check Brace

If the character at the cursor position is a left or right brace, E! will try to find the balancing right or left brace. An error message is issued if this brace couldn't be found in the text. Default assignment: **Ctrl+H**. See BEGINEND.EWD for an extension of this feature.

JoinLine

Join the current line and the following line. Default assignment: **Ctrl+J**. Lines are also joined when hitting Delete at the end of a line or BacksSpace at the beginning of a line.

Strip Flow

Remove all trailing spaces for the current text. This occurs automatically if the Strip Trailing Spaces option has been activated in the Local Options dialog box. Default assigment: **Ctrl+K**.

Load User File

If the word at the current cursor position is a valid filename, E! will immediately create an Editor and load that file into it. For example, if you are editing a C source file including a .H file, put the cursor under the filename and hit **Ctrl+L**. The file will be loaded.

User files are searched in the E! path defined in the Control Center Directories dialog box. Default assignment: **Ctrl+L**. This function can also be triggered by double-clicking on the filename with the right mouse button.

SplitLine

Split the line at the cursor position. Useful when the "Enter splits line" Mode is not activated. Default assignment: **Ctrl+S**.

Remove Tab / Add Tab

Remove or Add a tab position at the cursor position. Default assignment: **Ctrl+R** / **Ctrl+T**.

Delete Word

Delete the word at the cursor position. Default assignment: **Ctrl+W**.

Insert Line

Insert a new line in the text. If the "Enter splits line" option is On and if the Editor is in insert mode, this is done by splitting the current line at the cursor position. Otherwise an empty line is added after the current line. Depending on the editing context the cursor is moved to the most logical position on the new line (smart indenting). Default assignment: **Enter**.

Toggle Insert

Toggle between Insert Mode and Replace Mode. The cursor is always a vertical caret in Insert Mode and a horizontal caret in Replace Mode. Default assignment: **Insert**.

See also:

Block Functions

Block Functions

If text is selected in Block Mode, you can apply a set of useful functions to that block:

Fill Block

Fill the block with the specified character. Default assignment: **Ctrl+F3**.

Shift Block

Move all the text at the right of the left block margin to the right. The value of the shift to the right is defined by the block width. Default assignment: **Ctrl+F4**.

Sort Block

E! can sort lines very quickly. The sort key is defined by the marked area. If you have marked a text area in Line Mode, E! will use the whole lines to sort the marked area. If the text is selected in Block mode, on ly the substrings defined by the block width will be used as a sort key. Default assignment: **Ctrl+F5**.

Depending on the amount of lines to sort, the process may fail due to the lack of stack space. Generally, a second try will finish the sort since a part of the text will have been already sorted.

Add Block

If the marked Block contains numbers, these numbers will be added and the sum will be inserted at the cursor position. Don't forget that you can use the right mouse button to position the cursor while a selection is active. Default assignment: **Ctrl+F6**.

See also:

Editing Functions

Selecting Text and using the Clipboard

Selection Modes

E! offers 3 selection modes.

The Stream Mode is very similar to the selection mode found in most of the word processors and text editors. Selection begins at the current cursor position and includes all the text between this position and the end of the selection. While this type of selection is very useful for word processors it is not suitable for most editing tasks in a programmer's editor.

The Line Mode allows to select whole lines only. Wherever the selection begins or terminates only whole lines are included.

The Block Mode (also named Column Mode) allows to select a rectangular area in the text. E! provides many functions acting on a Marked Block such as Shifting, Adding, Moving, Sorting, Overlaying, etc...

See also:

Selecting Text using the Keyboard
Selecting Text using the Mouse
Clearing the Selection
Copying or Appending the Selection to the Clipboard
Inserting or Overlaying data from the Clipboard
Drag and Copy / Drag and Move
Copying and Cutting text without selecting

Selecting Text using the Keyboard

Stream Selection

You begin a Stream Selection with the keyboard by depressing the **Shift** key and using the **direction keys**, **PageDown**, **PageUp**, **Home** and **End** keys. The **Control** key can also be used. The selection terminates when you release the **Shift key**. A Stream selection involving only one line is always converted into a Block selection.

Line Selection

You begin a Line Selection by depressing the **Alt** key and using the **Up** and **Down** direction keys. Using the **Right** or **Left** direction keys selects the current line only. **Home** and **End** will select all the text from the current line to the top or to the bottom of the text respectively. Releasing the **Alt** key and depressing it again will begin a new selection.

Block Selection

You begin a Block Selection by depressing the **Control** and **Alt** keys simultaneously and by using the **direction keys** as well as the **Home**, **End**, **PageUp** and **PageDown** keys.

See also:

Selection Modes
Selecting Text using the Mouse
Clearing the Selection
Copying or Appending the Selection to the Clipboard
Inserting or Overlaying data from the Clipboard
Drag and Copy / Drag and Move
Copying and Cutting text without selecting

Selecting Text using the Mouse

Stream Selection

You begin a Stream Selection with the mouse by simply clicking on the left button and dragging the mouse cursor to the end of the selection.

Line Selection

You begin a Line Selection by depressing the **Alt** and **Shift** keys, clicking on the left mouse button and dragging the mouse cursor to the end of the selection.

Block Selection

You begin a Block Selection by depressing the **Control** and **Shift** keys, clicking on the left mouse button and dragging the mouse cursor to the end of the selection.

See also:

Selection Modes
Selecting Text using the Keyboard
Clearing the Selection
Copying or Appending the Selection to the Clipboard
Inserting or Overlaying data from the Clipboard
Drag and Copy / Drag and Move
Copying and Cutting text without selecting

Clearing the Selection

The selection is always cleared when you click in the text with the left mouse button (outside the selected area), when you hit a character or direction key or when you trigger an editing function that would conflict with an active selection. As mentioned above, the selection is not cleared if you change the cursor position using the right mouse button.

See also:

Selection Modes
Selecting Text using the Keyboard
Selecting Text using the Mouse
Copying or Appending the Selection to the Clipboard
Inserting or Overlaying data from the Clipboard
Drag and Copy / Drag and Move
Copying and Cutting text without selecting

Copying or Appending the Selection to the Clipboard

As usual the default key to copy the current selection to the clipboard is **Ctrl+Ins** (or **Ctrl+C**). **Shift+Del** (or **Ctrl+X**) will copy the selection to the clipboard and delete it from the text (Cut function).

E!, however, gives the clipboard more flexibility. Normally copying data to the clipboard removes the previously existing data. E! allows you to append text to the clipboard, adding the current selection to the text already present in the clipboard buffer. To do this simply use the Copy & Append or Cut & Append options in the Edit Menu of any Edit Window.

See also:

Selection Modes
Selecting Text using the Keyboard
Selecting Text using the Mouse
Clearing the Selection
Inserting or Overlaying data from the Clipboard
Drag and Copy / Drag and Move
Copying and Cutting text without selecting

Inserting or Overlaying data from the Clipboard

There are two ways of pasting the clipboard data into the text. The usual way is to insert the clipboard text at the current cursor position (or to replace the current selection if E! is configured to do that) by hitting **Shift+Insert** or **Ctrl+V**.

The other way, that will be useful in many situations is to Overlay the text at the cursor position by using the Paste & Overlay option from the Edit Menu. The Paste and Overlay feature doesn't work with Stream Mode selections.

E! always knows what kind of selection is currently in the clipboard. That is, it knows whether the clipboard text comes from another program (in that case it will consider it is a Line Mode selection) or if it comes from E! itself. In the latter case, it knows whether this is a Stream Mode, Line Mode or Block Mode selection and it will insert it in the text accordingly.

See also:

Selection Modes
Selecting Text using the Keyboard
Selecting Text using the Mouse
Clearing the Selection
Copying or Appending the Selection to the Clipboard
Drag and Copy / Drag and Move
Copying and Cutting text without selecting

Drag and Copy / Drag and Move

E! offers a feature that is unusual in text editors. Since always using the clipboard to copy and move data in your text may not be that productive, these transfers can be done by copying or moving the text directly, without using the clipboard.

Once you have made a selection, you can copy it to another place by simply clicking within the selection with the left mouse button. The mouse cursor will become a big "C" and you will be allowed to point to the location where you want the text to be copied. Release the left mouse button when you're done. This is a simple and fast way to copy text.

To move the selection instead of copying it, depress the **Control** key before clicking within the selection. This time the mouse cursor will become a big "M". If you release the left mouse button while the mouse cursor is located within the selection, nothing happens.

See also:

Selection Modes
Selecting Text using the Keyboard
Selecting Text using the Mouse
Clearing the Selection
Copying or Appending the Selection to the Clipboard
Inserting or Overlaying data from the Clipboard
Copying and Cutting text without selecting

Copying and Cutting text without selecting

If no selection is active, **Ctrl+Insert** or **Ctrl+C** will copy the current line into the clipboard while **Shift+Del** or **Ctrl+X** will cut it. This also works when activating these functions from the ribbon.

See also:

- Selection Modes
- Selecting Text using the Keyboard
- Selecting Text using the Mouse
- Clearing the Selection
- Copying or Appending the Selection to the Clipboard
- Inserting or Overlaying data from the Clipboard
- Drag and Copy / Drag and Move

Drag and Drop Support

E! supports the Drag and Drop feature of Windows 3.1. Depending on the location where filenames are dropped, E! will behave differently.

If you drop a filename on the Control Center (minimized or not) E! will immediately create a new Editor for that file or display its Edit Window if it is already edited.

If you drop a filename on a minimized Edit Window, the file will be inserted at the current cursor position in that Editor.

If you drop a filename on a restored Edit Window, the file will be inserted at the location pointed to by the mouse cursor when the left mouse button is released.

If you launch E! with filename parameters, only those files will be loaded in the new session. Files from the previous session will be ignored even if the Log Session option is ON. This is the case when you launch E! by dropping a filename on a Norton Desktop (tm) icon.

If you try to run a second instance of E! and if you pass it filenames on the command line, the second copy of E! will abort but before, it will pass the filenames to the running copy. So, this will have the same effect as if you had dropped the filenames on the Control Center icon.

Keyboard Assignment

Keyboard assignment is one of the most versatile feature of E!. Not only you can assign E! Editing Functions, Keyboard Macros, User Extension DLLs or character strings to any key but you can also have a different keyboard for each Editor.

You can define keyboards that are automatically loaded when you edit a file with a specific extension. You can also change the current keyboard while editing. If you have loaded a non-default keyboard while editing a file and if the Log Session option is activated, E! will remember which keyboard you have used with that file and reload it automatically when reloading the file.

Keyboards are defined in a single place: the Keyboard Assignment dialog box which is opened from the Local Menu of each Edit Window. This dialog box is rather self-explanatory but there are a few things you have to know to understand how the whole thing works.

Keystrokes can be assigned not only Editing Functions but also keyboard Macros or User-written DLL extensions. The type of assignment is selected by clicking on one of the relevant radio buttons. The listbox will be filled with the relevant data. **Please note that your DLL Extensions and Keyboard Macros must reside in the E! User Directory defined in the Control Center Directories dialog box**. Also, you don't have (must not) care about the file extensions. For your convenience, E! manages the file extensions for you. You just have to provide 8 characters names when loading or saving keyboards.

Once you have selected an object to be assigned to a keystroke, the current assignment, if any, will be displayed. Then you'll have to define the keystroke that will trigger that object. Go to the "First Key" field and simply hit the keys you want to be assigned. A string representing the key combo will be displayed in the field. If you want a two keystrokes assignment (for example to emulate Wordstar (tm) or Borland (tm) keyboards), click on the "Two Keystrokes Combo" check box and fill in the second field. When everything is correct, click on the Assign button.

Note: It is not possible to make an assignment to the ENTER key.

If this assignment is temporary, you don't need to save the keyboard before exiting the dialog box. To save your assignments in a keyboard file, just provide an 8 characters keyboard name and click on the Save button.

If you give a name which is actually a filename extension, this keyboard will be automatically loaded each time you edit a file with that extension. Otherwise, the default keyboard (if present on your disk) will be applied to the newly created Editor. If no default keyboard is present (DEFAULT.EWK), the standard built-in E! keyboard will apply.

To load a new keyboard while editing, select an existing keyboard from the Keyboard Name combo box and click on the Load button.

See also:

Keyboard Macros
User Sets

Keyboard Macros

Repetitive keyboard actions can be recorded either temporarily or in a macro file for later use. Recording keyboard macros is very simple. Hit **Ctrl+M** or select Begin Macro from the Edit Menu. Then do your work normally until you decide the macro has to beclosed. Hit the **Ctrl+M** key again or chose End Macro from the Edit Menu. The macro can then be replayed using the **Ctrl+Y** key.

If your macro must have a variable part, you can insert an Input Pause in the macro while recording it by hitting **Ctrl+I**. When replaying the macro, you will be given the opportunity to enter any character string. This string will be inserted in the text at the current cursor position. Then hit enter to continue macro execution.

If you begin a new macro, E! will ask you whether you want to save the current one. You can decide to save it by selecting the Save Macro option from the Edit Menu. As for keyboard files, just provide an 8 characters name. E! will take care of the file extension for you. Macro files have a .EWM extension.

To load a keyboard macro, select the Load Macro option from the Edit Menu.

Auto Macros are a special type of Keyboard Macros. Actually, their behavior depends on their name. When a file is loaded, E! will look in its User Directory for a macro with the same name as the file extension. If found, this macro will be executed against the loaded file immediately after loading. Auto Macros can be disabled from the Edit Options dialog box.

See also:

Keyboard Assignment

Syntax Expansion

Syntax Expansion is a feature allowing E! to bind a keyboard macro and a character string. Each time you will enter that character string followed by a space, the relevant keyboard macro will be played. For Example, this allows you to automatically enter "if then else" structures by merely typing "if" followed by a space.

Syntax Expansion sets are defined using the Syntax Expansion dialog box opened from the Local Menu. All available macros will be displayed in a listbox. You'll just have to provide the "key", that is, the string that is supposed to trigger macro execution. You must also specify for which filename extension the expansion is valid. So, you may define different Syntax Expansion sets for different source files. You may specify several extensions or a "*" character if the expansion should apply to all files.

You may also specify if the expansion is enabled or not. Depending on the editing context you may want to have several definitions for the same "key" and change between them while editing. E! will always use the first expansion that is enabled for the current extension.

Modifications to an existing entry are discarded if you don't click on the "Modify" button before exiting.

If you create an Expansion Set which name is a filename extension, this set will be automatically loaded for all files having that extension. Otherwise the Expansion Set contained in DEFAULT.EWX will be used if it exists.

See also:

User Sets
Keyboard Macros

User Menu

Since all editing functions are not present in the standard menu, you may want to make some of them appear in a customized menu. Moreover, you may want to have commonly used Keyboard Macros, Extension DLLs, Windows or DOS programs available in that user specific menu.

E! allows you to define a user menu that is global for all Edit Windows of one Editor. Menu items are described using the Define User Menu dialog box opened from the User Menu. Like in the Keyboard Assignment dialog box, you have to select the kind of object you want to assign to a menu item. Then you have to provide the menu title and the Windows or DOS command if necessary.

Modifications to an existing entry are discarded if you don't click on the "Modify" button before exiting.

If you create a User Menu which name is a filename extension, this menu will be loaded automatically for all files having that extension. Otherwise, the menu contained in DEFAULT.EWU will be used if it exists.

See also:

User Sets
Keyboard Macros
Keyboard Assignment
Syntax Expansion

User Sets

You may have noticed that some dialog boxes in E! share some functionalities. This is the case for the Keyboard Assignment dialog, the User Menu dialog, the Compiler Commands dialog and the Syntax Expansion dialog.

All these dialogs manage a collection of user defined data (menu items, keystroke assignments, etc...). Since these data are managed the same way, the corresponding dialogs offer you the same functions to manipulate them. These collections of data are called User Sets.

At the bottom of all these dialogs boxes you will find the same control area consisting of:

- A combo box displaying the name of the current User Set and allowing to select a new one

- A Load button

- A Save button

- An Exit button

- A Help Button

When an Editor is created, E! search the correct User Set using the following priority order:

1. E! tries to load the User Set (keyboard, usermenu, syntax expansion assignments, compiler command file) that was used for that file in a previous session.

2. If it doesn't find such a User Set, E! tries to load the User Set that has been defined in the "Associate" dialog box.

3. If this doesn't succeeds, E! tries to load the User Set that has the same name as the filename extension for that Editor.

4. If it doesn't find it, E! tries to load a User Set named "Default". If the default user set doesn't exist, E! uses its own built-in settings.

The name of the selected User Set appears in the Edit field of the combo box. When you open one of the mentioned dialog boxes, you are enabled to modify the User Set, add new items to it or remove existing items.

Your changes are reflected in all the Editors that are using the same User Set. When you exit these dialog boxes, E! prompts you to know whether you want to save the changes. You can use the Save button to save the changes in the current User Set. You may also change the current name or pick up the name of another existing User Set. Clicking on the Save button causes the current User Set to be saved in a file who has the name currently showing in the Edit Field of the combo box. The filename extension is added by E! automatically. Never add a filename extension yourself.

If you decide to exit the dialog box without saving your changes, these changes will apply to all Editors using that User Set until the very last of them has been closed. You can load another User Set at any time by using the Load button. As we have seen previously, changes are propagated to all Editors sharing the same User Set. However, if an Editor reloads the

same User Set from disk, it will be considered as having a new User Set. Be careful when saving User Sets with the same name, only the last one will be finally stored on your disk.

Let's look at a small example:

E! loads a file named MYPROG.PAS. It finds and loads PAS.EWK which is the keyboard that you have defined for .PAS files. All .PAS files edited by E! will share that same keyboard. If you modify the keyboard from one Editor, all other Editors containing a .PAS file will be submitted to the same modifications.

If you then reload the PAS keyboard from one of these Editors, it will now have the original PAS keyboard, not the modified one (unless you saved it in the mean time). Depending on what Editor is current when you use the Save button in the Keyboard Assignment dialog box, you will save a different version of the keyboard. This situation is not likely to happen very often but you should be careful when it occurs.

User Set files are either default files, extension based files or "normal" files. Since extension based files (files named after a filename extension) are automatically loaded only for the edited files having that extension, you'll have to decide a strategy for all the other files which you want to use the same User Set but that have a different filename extension.

1. You can load User Sets manually for each Editor. This is by far the most awkward solution.

2. You can make a copy of a User Set file and rename it with another extension (i.e. copy PAS.EWK to INC.EWK).

3. You can define Default User Sets including definitions for several extensions. This is possible for the Compiler Commands and the Syntax Expansion Sets only.

4. You can use the "Associate..." dialog box in the File Menu of each Edit Window to make associations between filename extensions and User Sets. This is the recommended solution.

See also:

Associations
Keyboard Macros
Keyboard Assignment
Syntax Expansion
User Menu

Associations

The "User Set Associations" dialog box allows you to bind a specific filename extension to one or more User Sets (that is, a Keyboard, a Compiler Command Set, a Syntax Expansion Set or a User Menu).

The dialog box contains 5 edit fields. The first one displays the current filename extension. Change it if you want to process another filename extension. The other edit fields show you the current values (assignments) for the Keyboard, the User Menu, the Compiler Command Set and the Syntax Expansion Set respectively.

You can enter new values manually or select an existing file (User Set); the combo boxes attached to each field will show you which User Sets are available in the User Directory.

Click Ok to accept the values. These settings are stored in EW.INI. Although you may change them manually, it is recommended that you use this dialog box to make the changes.

Now, if you load a file with that filename extension, E! will try to load the User Sets that you have specified.

Warning: Be aware that files with the same filename extension that were edited before you change these settings were registered as users of either the default User Sets or the Users Sets that you had loaded for them manually.

If you reload these files, they will use these User Sets, not those that you have just defined. See User Sets for the priority order used by E! when searching User Sets. Since this registration will disappear after a number of editing sessions specified in the Usage variable of EW.INI, this is not a big problem unless you reload them immediately. Otherwise set the User Sets manually for these files.

See also:

User Sets
Keyboard Macros
Keyboard Assignment
Syntax Expansion
User Menu

Searching

Standard Functions

A new search is always launched with **F5** (default assignment). A dialog box opens and prompts you for the search string. If you intend to run a Search and Replace process, just check the Replace checkbox and fill in the "Replace with" field. Please notice that you can retrieve strings previously entered in those fields by simply using the combo boxes attached to each of them. E! remembers up to 10 search strings. This value can be changed by modifying (or creating) the "history" parameter in the [ewcc] section of EW.INI.

You may have some strings or regular expressions that you use repeatedly in search operations. Normally these expressions are lost when they reach the end of the history list. If you want them to be permanently inserted in the history list, proceed as follows:

1. Open EW.INI

2. Goto to the [searchopt] section (or create it if it doesn't exist).

3. Create a "spatterns=" entry for the strings that you want to appear in the "find" combo box. Add your strings to this entry. Since these strings may contain spaces, you will have to use a caret (^) as a separator. If a caret should appear in one of those strings "escape" it with the backslash () character. It will then be ignored. The Escape character () will be maintained in the string though. Since this is useful mainly for boolean search operations, this will not be a big problem because the regular expression parser will eliminate the Escape character when necessary.

4. Proceed similarly for the replacement strings by creating a "rpatterns=" entry in the same section of EW.INI.

The Search dialog box gives you an opportunity to change the Search Options. Search Options are global. If you change them from any editor or from the Control Center, the new values will apply for all editors.

Click OK to launch the Search. The found string will be automatically selected (not only highlighted). If you have selected Search and Replace mode, a dialog box will be opened and will prompt you whether you want to replace the found string, skip it, replace all further occurrences of that string without prompting or stop the process.

Once you have launched a Search process, you can repeat the same search with **Shift+F5** (default assignment).

E! also has a nice feature allowing to automatically find the next occurrence of the word at the cursor position. Just hit **F6** to activate the search while the cursor is located within a word. E! will look only for words not for strings imbedding that word.

If you have selected Regular Expression Search in the Search Options dialog box, E! will perform a Boolean Search. Regular Expressions are a very powerful tool but are sometimes a little tricky to use.

See also:

Boolean Search
Sample Regular Expressions
Boolean Search and Replace

E! Specifics
Search Options

Boolean Search

Regular expressions are a convenient way to search complex or "loosely defined" expressions in a text. For example, you may want to search all strings beginning with a "B" and ending with a "E" and including any number of characters. Or any occurrence of a word at the end of a line. Or even all the valid identifiers for the language you're using.

Regular expressions allow you to achieve this. They use a special "search language" to define the search string.

During a regular expression search certain characters in the search string have a special meaning:

? : REPLACE ANY CHARACTER

? in the search string will match any character.

/al?/

will match "all" and "ale".

? replaces only one character. You can use as many "?" as you want in the search string.

% or < : MATCH THE BEGINNING OF A LINE

"%" or "<" at the beginning of the search string specifies that the matched string has to be located at the beginning of a line. "%" and "<" used at any other location in the search string have no particular meaning.

$ or > : MATCH THE END OF A LINE

"$" or ">" at the end of the search string specifies that the matched string has to be located at the end of a line. "$" and ">" used at any other location in the search string have no particular meaning.

[..] or [..-..-..] : CLASS DEFINITION

The characters enclosed within the square brackets form a class. You can think of a CLASS as a restrictive "?". Any character belonging to the class will match. There are two ways to specify the characters belonging to the class. You can describe each single character like in [aeiou] or you can define ranges like in [a-z0-9] which means all characters between (and including) a and z and all characters between (and including) 0 and 9. Both methods may be mixed like this: [aeiou0-9] that is equivalent to [aeiou0123456789].

When defining a range in a class the first character must precede the character after the dash in the ascii set order.

A right square bracket following the right square bracket of the class definition specifies a "[" effectively. The following characters encountered within a class have no special meaning:

< > % $ * + @ # # _ { } ?

(also true for "^" except when it is the first character in the class and for "-" when the last character in the class).

A class always matches ONE character whatever the number of characters specified in the class.

[^..] : CLASS NEGATION

A caret (^) as the first character of the class specifies that the class is made up of ALL characters EXCEPT those defined in the class.

* : MATCH 0 OR MORE OCCURRENCES OF PREVIOUS CHARACTER (short match)

"*" specifies a match with 0 or more occurrences of the character previously defined in the search string. This may be a unique character of a class. In that case E! looks for the shortest possible match.

Example:

"ba*b" will match "bb", "bab", "baab", etc...
"b?*b" will match "bb", "bxb", "byyb", etc...

Considering the last example, if the text contains the following sequence:

"baaaabxxxxxbzzzzzzzzb"

the match will occur with the shortest string matching the regular expression that is:
"baaaab" and not "baaaabxxxxxb" or "baaaabxxxxxbzzzzzzzzb"

+ : MATCH 1 OR MORE OCCURRENCES OF PREVIOUS CHARACTER (short match)

"+" behaves exactly like "*" but at least one matching character is mandatory.

@ : MATCH 0 OR MORE OCCURRENCES OF PREVIOUS CHARACTER (long match)

: MATCH 1 OR MORE OCCURRENCES OF PREVIOUS CHARACTER (long match)

"@" et "#" behave like "*" and "+" but E! searches the longest possible match. In the preceding example the match would occur with "baaaabxxxxxbzzzzzzzzb".

: ESCAPE CHARACTER

"" is used to disable the interpretation of special characters. You will use it to tell E! to ignore the special meaning assigned to all characters used to specify a regular expression:

< > % $ * + @ # # _ { } ? ^ -

To insert a "" in the search string use "\".

Regular expressions usually recognize "" or "" that specify a tab character or a carriage return. A line in E! never contains such characters. So and will yield "t" and "n" respectively.

_ : (Underscore) SPECIFY CURSOR POSITION

Once E! has found the string specified by the regular expression, it will locate the cursor under the character immediately following the "_" in the regular expression.

Examples:

"a _< b" searches "a < b" and puts the cursor under "<" when the string is found.

"if _[^]* then" searches for an "if then" structure and locates the cursor at the beginning of the expression defining the condition.

This feature is very convenient to execute a macro from a specified location within a string.

The underscore character "_" may also be "escaped with "".

See also:

Standard Search Functions
Sample Regular Expressions
Boolean Search and Replace
E! Specifics
Search Options

Sample Regular Expressions

To help you understand regular expressions, here are some examples of classical expressions.

[a-zA-Z][a-zA-Z0-9_]#$ or [a-zA-Z][a-zA-Z0-9_]#>

This expression specifies any valid PASCAL identifier located at end of lines.

< #[a-zA-Z][a-zA-Z0-9_]# or < #[a-zA-Z][a-zA-Z0-9_]#

Ditto for PASCAL identifiers located at beginning of lines or preceded by blanks.

myfile.[^]*

Searches any filename beginning with "myfile" with any extension.

<BEGIN or %BEGIN

Searches all BEGIN keywords at the beginning of a line.

<END;>

Searches "END;" alone on a line.

$[0-9a-fA-F]@

Searches an hexadecimal number in a Turbo Pascal program.

<$

Searches any line beginning with a "$".

([^():<>]*=[^()]*)

Searches assignment statements in PASCAL.

See also:

Standard Search Functions
Boolean Search
Boolean Search and Replace
E! Specifics
Search Options

Boolean Search and Replace

You can specify in the search string which parts of it can be reused in the replace string. Let's illustrate this with an example.

Let's assume you are searching all expressions in the text where an identifier beginning with "VAR" is involved in an equality assignment with "CONST". Something like

VAR?*= *CONST

Let's also assume that you want to replace the equal sign in all these expressions with "<>". How can you specify that only the equal sign has to be changed and that the rest of the expression should remain unchanged ?

You will use subgroup specifiers, that is, curly braces (), like this:

{VAR?*}={ *CONST}

(1st group: VAR?* - 2nd group: *CONST)

These groups will be referred to in the replace string using "" where n is the number of the group. Up to 10 groups may be defined ranging from to .

So, the replace string will be:

<>

This way, will stand for "VAR?*", will stand for " *CONST" and only the equal sign will be replaced with "<>".

VAR_ONE = CONST

would be replaced with

VAR_ONE <> CONST,

and

VAR_TWO = CONST

would be replaced with

VAR_TWO <> CONST and so on...

Subgroup specifiers cannot be nested. They can be "escaped" using "".

The expression must contain the same number of right and left "non escaped" curly braces.

If the replace string must contain a "" followed by a digit and if this sequence is not intended to be a reference to a subgroup, you can escape the sequence as follows:

"\3".

See also:

Standard Search Functions
Boolean Search
Sample Regular Expressions
E! Specifics
Search Options

E! Specifics

As mentioned above "\t" and "\n" have no special meaning.

The regular expression parser is designed to be compatible with any other search option except backward search. The Search Options dialog box automatically handles option inconsistencies.

Using regular expressions is not that easy but they are a very powerful tool. Be aware that a regular expression search is much slower than a simple search. This is a highly recursive process. Specifying very general regular expressions may lead to a ve ry long search time. Try to specify significant expressions allowing the interpreter to select strings more easily.

For example, "?*anytext" would lead to catastrophic results because a lot of strings would match this regular expression.

E! is a line-oriented editor. So, you can't specify expressions matching strings across line boundaries.

See also:

Standard Search Functions
Boolean Search
Sample Regular Expressions
Boolean Search and Replace
Search Options

Grep Utility

GREP is a common utility that is able to search which files on your disk contain a given character string. E! has a built-in GREP allowing to search and edit the files in the same process. Found files are opened at the exact location in the text where the first occurrence of the string was found. The E! built-in GREP also supports boolean (regular expression) search. The accepted syntax for a regular expression is the same as described in the Boolean Search section.

The Grep Dialog Box

Find

Enter here the string or regular expression to be searched within the specified files.

FileSpec

Enter here the the file specification defining the files to be searched.

From Directory

This is the directory where Grep will begin to search the files.

Boolean Search

Check this option if you have specified a regular expression in the Find field.

Ansi String

Check this option if the find string is "ANSI sensitive".

Case Sensitive

Check this option for a case sensitive search.

Search Subdirectories

Check this option to search all subdirectories starting from the directory specified in the From Directory field.

All the files matching the given specifications will be listed in the Found Files listbox. You may then Edit them or Remove some of them from the list.

ANSI vs. OEM

This topic is important mainly for non english speaking users. The differences between the ANSI character set used by Windows and the ASCII (OEM) character set commonly used under DOS may lead to problems when loading or saving files containing accented vowels or semi-graphics characters.

There is no simple solution to that problem. We have tried to give you all the tools necessary to manage your files easily. Here is a description of the way E! handles this problem.

Within E!, the character set used to represent the loaded file depends on the font that is currently selected. If the font is an ANSI font all text lines are translated to ANSI. If the font is an OEM font, all the lines are translated to OEM. This is done transparently when you select a new font.

The Local Options dialog box of the Control Center allows you to decide whether, by default, the loaded file will be considered as ASCII and therefore converted to ANSI if the current font is an ANSI font. This information is given to E! through the LoadAscii checkbox. Likewise, the SAVE ASCII checkbox specifies whether the saved text has to be translated into ASCII anyway if it is currently edited with an ANSI font.

So far, so good. The problem is more difficult to solve when you want to load a file as ANSI while the Local Defaults are set to ASCII. Since we cannot set Local Options before the file is loaded, E! has no direct way to know how it should load the file unless you change the Local Defaults in the Control Center (kind of a chicken-egg problem). This method is rather cumbersome. We have tried to give a simple solution to this problem.

If you want to load a file that cannot accept the current default settings don't use the Control Center listbox to load it but rather open an Open dialog box from an Edit Window. This dialog box has a "Load Ascii" checkbox that temporarily overrides the current settings. If no Edit Window is opened, you have to change the settings from the Control Center and to change them again (if necessary) after the file has been loaded.

Once the file has been loaded you can set the Load Ascii checkbox according to your needs. This setting will be used when Refreshing the file using the File | Refresh command or when inserting a file using the File | Get File command. Moreover, the information will be stored in EW.INI and taken into account if you reload the file during a future session (provided the Log Session feature is On).

When saving the file using the Save option of the File Menu, the setting of the Local Save Ascii checkbox will be used. If you use the Save As option, the dialog box will allow you to specify whether the file should be saved as Ascii.

You should also be aware that the Draw function of E! cannot work if the current font is an ANSI font. Semi-graphics characters used to draw boxes cannot be saved in ANSI mode. E! will refuse to activate the Draw mode when an ANSI is selected font but cannot prevent you to save the file in ANSI mode even if the file has used non ANSI characters.

Drawing

The Draw Mode allows you to build boxes within your text very easily. You just have to activate the Draw Mode from the Edit Menu or from the ribbon and to select a Draw Mode.

While in Draw Mode use the direction keys to "draw" your object. E! selects the correct semi-graphics characters automatically. Drawing is undoable like any other function.

If you want to temporarily suspend the Draw mode, hit the ScrollLock key. Hit that key again to resume Draw mode. The Draw Mode icon of the Ribbon will change its color accordingly (the icon is red while Draw mode is suspended).

Use Mode 9 (blanks) to erase a drawing.

Select the Draw Mode again to end drawing.

You can record a drawing as a macro. However, before replaying the macro, you have to set the drawing mode using the Draw Mode dialog box.

Printing

Printing a file in E! is as easy as with any other Windows program. The steps for printing are as follows:

1. Verify that you have selected the correct printer using the Printer Setup option from the file menu.

2. Verify the page Setup using the Page Setup Menu. Margins are specified in columns and lines. Header and Footer can be defined using the following syntax:

Headers and Footers may contain one of the following characters preceded by a '%' to indicate a special function:

#	current page number
T	current system time
D	current system date
F	current filename
<	left justify
>	right justify
$	"Push" character
[	left justify on odd pages, right justify on even pages
]	right justify on odd pages, left justify on even pages
other char.	that character

The Push character allows you to have a part of the string left justified and another part right justified. The Push character acts as a spring that would enlarge the string from the Push character position until the maximum allowed length has been reached. So, the part of the string preceding the Push character will be left justified and the part of the string following the Push character will be right justified. In that case all other justification characters inserted before the Push character are ignored. If you insert a justification character after the Push character, its effect will be cancelled. The basic rule is that the last inserted justification character prevails.

This syntax is compatible with the LISTT program provided by Borland (tm) with early versions of their compilers.

3. The Page Setup dialog box also allows you to select a printer font. The "fixed-pitched only" option allows you to select non proportional fonts only. You can print a file using any font but remember that E! can only use fixed-pitched fonts to display the text in a window.

4. Select the Print command.

Comparing

If you want to compare two files, proceed as follows:

1. Go to the Control Center, make sure these files are currently edited and select both files in the Select listbox. Be aware that the comparison will begin at the current line in each file. You may want to position the cursor at the right position before proceeding.

2. Click on the Compare button.

3. E! will show you both files at the location of the first difference it has detected. For your convenience, both Edit Windows will be automatically tiled.

4. If you want to continue the comparison, position the cursor on two identical lines in each text and select Compare from the Bookmark Menu. You don't need to go to the Control Center again.

Compiling

From E! you can compile any program using any DOS Compiler provided this compiler will run into a Windows DOS box. Most compilers are supported automatically and E! will be able to analyze their output and to point to compiling errors in the program source directly.

Command Sets

Setting up E! for a standard compiler is an easy task. You just have to open the COMMANDS dialog box from the Compile Menu and to fill in the different fields. At least one Compile command is necessary but you are not obliged to provide a command in each field (for example, you can provide a Compile and a Make command leaving the Run, Rebuild and Debug command fields empty).

You can provide batch names if you want. 4DOS (tm) batches (.BTM) are also supported.

Enter commands exactly as you would enter them on the DOS command line. Don't care about redirection. E! will take care of this for you.

You may define several commands for the same set of filename extensions. E! will use the first set of commands that is enabled (Enabled checkbox). The Enabled checkbox makes it easy to manage different compiler commands for one set of filename extensions.For example you can define compiler commands for debugging the program and other commands for the production version. Just enable the commands you want to use (don't forget to disable the ones you don't want).

Modifications to an existing entry are discarded if you don't click on the "Modify" button before exiting.

As with Keyboards, Syntax Expansion Sets and User Menus, E! will try to load the compiler command file that has the same name as the filename extension. Otherwise it will load DEFAULT.EWC if it exists.

You can chose between two strategies when creating compiler command files. You can create a unique DEFAULT.EWC file containing all compiler definitions for all filename extension sets that you are supposed to use.

You may also define separate command files containing compiler commands only for one set of extensions.

There is a small difference between both strategies. If you define a separate set of commands for one specific extension, E! will load that set of commands only for the files with that extension. Other files will inherit the default Command set. This may lead to some problems because Compile commands will not be enabled for those files until you explicitly load the appropriate Command set. One work-around to this problem is to make a copy of the original Command file, giving it the name of another extension. For example, if you have defined a Command set in C.EWC, you can make a copy of C.EWC in H.EWC. This way all .C and .H files will use the same set of commands.

However, we recommend using separate Command sets for compilers generally involving a single type of file like resource compilers, Lint programs or macro compilers. Be also aware that the program triggered by the Compile Command set doesn't have to be a compiler. You can use all the Compiler Support feature to trigger any process and use the Message Descriptor to generate messages based on the program output. To be interpreted by E!, at least one line from the compiler output stream should display the filename and a line number.

See also:

Project File

Before beginning the compilation process make also sure that you have created a project file for your program. The project filename defaults to EW.PRJ but you can change it in the Commands dialog box. It is a simple ASCII file containing the name of all the source files involved in the compilation process. E! absolutely needs this file if the compilation process involves more than one source file.

Please enter only one file specification per line. Wildcards are allowed. Lines beginning with a double asterisk are considered to be comments. If you didn't specify a full pathname for the project file, this file must reside in the same directory asyour source files.

You may share a unique project file between several compilation processes. E! needs to have at least the names of all the files involved in the compilation process but there can be more in that list. So, a single project file may contain filenames for several different projects. Be aware however, that big lists will consume memory (which will be released after the compilation process terminates). Be especially careful when including many filespecs containing wildcards.

It is not necessary to have a project file if the compilation process involves only one file (the file from which the compilation has been launched). Otherwise E! will need a list of all involved source files to analyze the compiler output. If an error occurs in a file that is not listed in EW.PRJ, E! will not load the file in error but will display the compiler output. E! also displays the compiler output when the compilation process succeeds or if the error is not related to a source file.

See also:

Command Sets
Compiling Process
Non Standard Compilers

Compiling Process

When the Compile command is selected, E! opens a DOS window and runs your compiler. The DOS window will be shown only if the "Show DOS Job" option has been checked. The default behavior is to show the compiler job in a DOS window. If you want that compiler job to be shown full-screen, please modify EWX.PIF accordingly.

The compiler will do its job normally but E! will install a "spy" in the background. This small program (EWX.EXE) will redirect all the compiler's output into a file named EWX.OUT. This will not prevent the compiler from displaying its output on the screen too. Once the compiler is done, E! will read the contents of EWX.OUT and look for compiling errors. If any error is found, E! will load the relevant file (if not currently edited) and point to the error. You can switch from error to error by using the Next / Previous Error commands from the Compile Menu.

See also:

Command Sets
Project File
Non Standard Compilers

Non Standard Compilers

If E! fails to recognize your compiler output, this is very likely a problem with the format of the error messages issued by this compiler. E! gives you the opportunity of describing the format of these messages using the syntax given below. The format description has to be entered in the Msg. Descriptor field of the Commands dialog box.

E! needs at least three values to automatically display the source code in error:

1. The name of the source file. This is retrieved automatically in any case provided the compiler includes that name in the error message. Otherwise nothing is possible.

MIND: If your compiler doesn't insert the filename of the source file in the error message, E! cannot locate the error. This may sound strange but some compilers have this behavior.

2. The row number of the line in error.

3. The error message itself.

Additional information can be displayed or used if provided by the compiler:

1. The column number of the item in error

2. The error code

E! uses a tiny macro language to allow you to describe the error message format. This language uses five keywords:

<ROW>	Placeholder for the row number
<COL>	Placeholder for the column number
<ERR>	Placeholder for the error code
<MSG>	Placeholder for the message itself
<TYP>	Placeholder for the message type (generally ERROR or WARNING)
<VAR>	Placeholder for any other variable part including the filename.

A small example will make things easier.

Let's assume your compiler issues this kind of message:

*** ERROR 202 IN LINE 26 OF C:C51PEXAMPLEFIB.C: syntax error near '}'

Here is how to build the Message Descriptor for that compiler:

The Message Descriptor will look very much like the message above but some items will be replaced by the appropriate keywords. The rules are as follows:

- the string defining the line in error is replaced with <ROW> (mandatory)

*** ERROR 202 IN LINE <ROW> OF C:C51PEXAMPLEFIB.C: syntax error near '}'

- ditto for the column: <COL> (not relevant here)

*** ERROR 202 IN LINE <ROW> OF C:C51PEXAMPLEFIB.C: syntax error near '}'

- ditto for the error code: <ERR>

*** ERROR <ERR> IN LINE <ROW> OF C:C51PEXAMPLEFIB.C: syntax error near '}'

- the error message part itself is replaced with <MSG> (mandatory)

*** ERROR <ERR> IN LINE <ROW> OF C:C51PEXAMPLEFIB.C: <MSG>

- the string indicating the error type (ERROR or WARNING) is replaced with <TYP>

*** <TYP> <ERR> IN LINE <ROW> OF C:C51PEXAMPLEFIB.C: <MSG>

- any other variable string is replaced with <VAR>, including filename (mandatory)

*** <TYP> <ERR> IN LINE <ROW> OF <VAR>: <MSG>

- any fixed string is left unchanged (mandatory but in some cases you could replace the string with <VAR>)

*** <TYP> <ERR> IN LINE <ROW> OF <VAR>: <MSG>

The general structure of the message must be maintained as well as any blank and punctuation sign when not included in a variable part. For example the following descriptor for the above message will cause E! to fail in the message analysis:

*** <TYP> <ERR> INLINE <ROW> OF <VAR>: <MSG> (missing blank before LINE)

Leading and trailing blanks are discarded anyway. So, you don't have to care about them.

The use of <VAR> is restricted to the parts of the message that are really variable. You could use it for fixed parts, for example like this:

<VAR> <TYP> <ERR> IN LINE <ROW> OF <VAR>: <MSG>

but this would give the parser additional work. The more precise the descriptor is, the faster the messages are processed and the lower the chances are to fail.

So, the correct Message Descriptor will be

*** <TYP> <ERR> IN LINE <ROW> OF <VAR>: <MSG>

Once again, since E! supports all popular compilers automatically, the use of a Message Descriptor is very rarely needed. However, if you are not satisfied with the standard behavior of E!, you may decide to use a Message Descriptor even if it is not needed. This will allow you to modify the contents of the error message displayed in the Edit Window when using Next/Previous Error.

See also:

Command Sets
Project File
Compiling Process

About the Current Directory

Under Windows the idea of current directory is somewhat foggy. Windows is not a real multitasking environment. It is based on DOS which cannot maintain a copy of its environment for any running Windows application. Only the 386 enhanced mode DOS windows provide separate DOS environments. Even in 386 enhanced mode, all Windows applications belong to the same system task.

So, when you switch from one Windows application to another you don't know what happens to the base DOS environment. More precisely, you don't know when the application changes the current directory. For example, only a few users are aware that manipulating a directory listbox, that is, selecting a directory from a listbox, makes that directory the current directory. Most of the time, you don't know "where you are".

Like many other applications, E! (that is, actually, Windows) makes the last selected directory from an Open File listbox the current directory. But what happens if you switch to the Control Center without using the directory listbox? E! always remembers which drive and directory were current when you switched to another application. As soon as the Control Center gets the focus again, that drive and that directory are made current.

Also, when triggering a compilation, E! makes the directory where the source file from which the compilation is run resides, the current directory in the DOS window opened by the compilation process. This way, everything will run the same way as if you had launched the compile command from the DOS prompt in that directory.

AntiVirus Protection

E! as some kind of antivirus protection. This protection is disabled by default because this makes the loading process a little longer. If you want to activate this feature, just add the following statement to the [system] section of EW.INI.

[system]
autocheck=yes

Create this section if it does not exist.

If autocheck is activated, E! will check its own CRC when loading. This takes a few seconds but this will make sure that EW.EXE has not been modified. If anything is wrong, E! will complain and exit. If you were to decide to edit E! resources using tools like Resource Workhop (tm) or Resource Toolkit (tm) the CRC will be changed and you'll not be able to load EW.EXE any more if autocheck is on.

Autocheck will always be ON until the program has been registered.

Modifying EW.INI

Normally, you should not modify EW.INI manually. Since the Log Session process is rather complicated you should not try to modify lists maintained by E! within EW.INI.

However, some rarely changed options are not available from any dialog box. The only way to change them is to edit EW.INI.

[ewcc]

history=n

where n defines the history length in combo boxes retaining information from one session to another (i.e. combo boxes of the Search dialog box).

[system]

resleftfree=percent

One of the biggest problems a Windows application has to face is System Resource shortage. Although Resource management has been somewhat enhanced under Windows 3.1, System Resources are still very limited. E! Edit Windows consume System Resources. Practically it will be difficult to open more than 15 or 16 Edit Windows. E! refuses to create a new Edit Window when the percentage of free System Resources goes under the value defined by "percent". This value defaults to 8.

autocheck=yes/no

Specify whether E! must perform CRC autocheck on EW.EXE when loading.

alternatehelp=filename

This help file will be used when the **Shift+F1** is pressed while the cursor is under a keyword. The default filename for the **Alternate Help File** is "win31wh.hlp", the Windows SDK help file.

undotruncate=yes/no

The data used by the Undo function are created by each editing function and stored either in memory or onto the disk. Each Editor has its own "spill file" where the data needed to undo editing functions are stored. Normally this file keeps growing until the corresponding Editor has been removed from memory. This is not a problem if you have enough disk space available (the spill file is deleted when closing the Editor). However, if you want to save some disk space while editing, E! can use a more clever scheme.

You have seen that the number of Undo actions that are pushed on the Undo stack is limited by the Undo Levels variable. When this level has been reached, the oldest actions on the Undo stack are discarded but not the corresponding data which remain in the spill file.

If you set undotruncate to yes, E! will take the time to delete the data corresponding to discarded Undo actions. This way, the size of the spill file will be limited to the size of the Undo information that can be actually undone.

Spill file operations take time. So, the drawback of this feature is that, depending on which

Editing functions you are currently using, the disk activity may increase significantly. This will not be a problem if you have a fast hard disk but we don't recommend to use this feature if your disk is slow or if you have enough disk space available.

[editopt]

loadbuffersize=n

where n is the size of the temporary memory buffer created when loading a file. This value defaults to 4096 bytes. On some systems, performance can be enhanced when increasing this value.

[fileopt]

usage=n

E! is able to remember all the local options settings for one specific file and to reactivate this options when loading this file in a future session. However E! cannot keep these data on the disk permanently: EW.INI would grow too big. The strategy used by E! is as follows: if the file has not been reloaded during the next n sessions, data specific to that file will be erased from EW.INI. n defaults to 3.

timeout=sec

When E! launches a compilation, it waits for the compilation process to terminate before it begins analyzing the compiler output. If for any reason the DOS job never terminates (i.e. your compiler has a bug and the DOS window locks up), E! would have to wait forever. The timeout option defines the maximum time a compilation process may last. The default is 600 sec.

[searchopt]

rpatterns=
spatterns=

Please see the SEARCHING section for a description of these entries.

All other entries should not be changed manually.

Extension DLLs

Extensions DLLs are user-written DLLs using the E! API. These DLLs can extend the basic functions fo E! by triggering E! functions or by setting hooks in the code of E! to modify the behavior of these functions. Extension DLLs can be written using any language.

User Extension Dialogs

Registered users will receive the EW API Programmer's Guide explaining how to use and write Extension DLLs

---

See also:

User Extension Dialogs

User Extension Dialogs

3 Dialog Boxes are available to Load, Unload and Execute Extension DLLs. In each of these dialog boxes, a listbox will display a list of DLLs. The list will be built according to the following rules:

Only .EWD files stored in the User directory are listed.

DLLs that are already loaded are also listed in the Load dialog box but nothing will happen if you try to reload them.

DLLs that do not export an EWExecute function will not show up in the Execute dialog box.

Only loaded DLLs will appear in the Unload dialog box.

You don't need to load a DLL already appearing in the User Menu. Clicking on the menu item will cause the DLL to be loaded automatically.

More information on how to use Extension DLLs will normally be available from the authors of these DLLs.

See also:

User Extension

Keyboard Index

See also:

Command Index

Editing Functions
Block Functions

See also:

Keyboard Index
Procedure Index
Help Features

Procedure Index

See also:

Keyboard Index
Command Index
Help Features

Help Features

You can get help at any time by hitting the **F1** key.

You can also display a reminder message for the ribbon buttons by clicking with the right mouse button on the ribbon button about which you want information.

Shift+F1 is the **Alternate Help** key. If you hit this key while the cursor is on a keyword, the **Alternate Help File** will be loaded and the keyword will searched in that file.

By default, the **Alternate Help File** in the Windows 3.1 SDK help file. You may change this by modifying EW.INI.

See also:

Keyboard Index
Command Index
Procedure Index

Copyright an Licence Agreement

E! for Windows

For ordering information, please see the file ORDER.FRM.

License Agreement and Warranty Disclaimer

You should carefully read the following terms and conditions before using this software. Use of this software indicates your acceptance of these terms and conditions. If you do not agree with them, do not use the software.

Non-registered Version

The Non-registered Version of E! for Windows is a version to which the Registration procedure has not been applied. To run the registration procedure you need a registration code which will be sent to you after payment. This procedure disables the reminder dialog box and allows you to disable the CRC checking feature of E!.

You are hereby licensed to: use the Non-registered Version of the software for a 60-day evaluation period; make as many copies of the Non-registered Version of this software and documentation as you wish; give exact copies of the original installation diskette to anyone; and distribute the Non-registered Version of the software and documentation in its unmodified form via electronic means. There is no charge for any of the above.

You are specifically prohibited from charging, or requesting donations, for any such copies, however made and from distributing the software and/or documentation with other products (commercial or otherwise) without prior written permission, with one exception: Disk Vendors approved by the Association of Shareware Professionals are permitted to redistribute E! for Windows, subject to the conditions in this license, without specific written permission.

However, Disk Vendors are not allowed to charge more than $5 (or its equivalent in their local currency) for a disk containing the Non-registered Version of E! for Windows. Moreover, E! for Windows may not be distributed on diskettes containing other products, even in compressed form.

Unregistered use of E! for Windows after the 60-day evaluation period is in violation of federal copyright laws.

Registered Version

The registered version of E! for Windows as well as the API documentation may not be distributed/uploaded in any form and for any reason by individuals or Disk Vendors. Violations of this agreement will be prosecuted.

Evaluation and Registration

This is not free software. This license allows you to use this software for evaluation purposes

without charge for a period of 60 days. If you use this software after the 60-day evaluation period a registration fee is required (please see ORDER.FRM). Payments must be in US dollars or in german Marks (DM) and should be sent to the address mentioned in ORDER.FRM. Credit card ordering and quantity discounts are available.

When payment is received you will be sent a registered copy of the latest version of E! for Windows, a printed documentation and the E! for Windows API documentation. You will also benefit from a fax hot-line service at the following fax number in Germany:

(0721) 373842 (please give your registration code)

(You may also get support on Compuserve - IBMAPP EDITORS FORUM)

One registered copy of E! for Windows may be dedicated to a single person who uses the software on one or more computers or to a single workstation used by multiple people.

You may access the registered version of E! for Windows through a network, provided that you have obtained individual licenses for the software covering all workstations that will access the software through the network. However, E! for Windows has not been tested with all networks, so we cannot guarantee that this will be possible in all cases.

Governing Law

This agreement shall be governed by the German law.

Disclaimer of Warranty

This software and the accompanying files are sold "as is" and without warranties as to performance of merchantability or any other warranties whether expressed or implied. In particular, there is no warranty for the optional virus scanning (CRC verification) feature. Because of the various hardware and software environments into which E! for Windows may be put, no warranty of fitness for a particular purpose is offered.

The user must assume the entire risk of using the program. Any liability of the seller will be limited exclusively to product replacement or refund of purchase price.