
WinIDE Integrated Development Environment - User Guide v.1.00

1 WinIDE Overview

WinIDE is an editing application that allows seamless integration of several different programs into one development environment. This text editor allows the user to launch different programs including assemblers, compilers, debuggers, programmers, and more. If configured correctly, the environment can also read back errors from these programs and display them for the user. Having a development environment which consists of several separate programs allows the user to replace any part of the environment. For instance, instead of using PEmicro's assembler, a third party C-compiler can be used. If error detection is set correctly, the user's errors will be highlighted and the compiler's error message displayed. The user never has to leave the editing environment to run the compiler, and the editor automatically opens any files which contain an error.

In addition, WinIDE contains useful features such as color-coding of text based on file type for easier scanning, and project management facilities which allow you to access files and move them in and out of your project with ease. WinIDE gives you fine control over a variety of settings so that you can configure the environment to best suit your needs, enabling you to work with comfort and efficiency.

2 Environment Configuration

WinIDE allows you to customize many environment settings so that you can create a working environment that suits you. The user may modify these settings by selecting options from dialog boxes using the Environment pull-down menu. Among the selections listed in the pull-down menu are:

- **Environment Settings** – Opens a dialog that includes global settings for the assembler/compiler and other executables, as well as save options, file type filters, backup settings, etc. See **Section 3 - Environment Settings Dialog**.
- **Editor Options** – Opens a dialog that includes syntax-specific settings, such as the ability to fine-tune color formatting and set text-level options, like word wrap and indenting. The dialog also allows you to set hotkeys for various functions. You must have a file open for this dialog to be available. See **Section 4 - Editor Options**.
- **Syntax Preferences** – Opens a dialog that allows you to choose from color formatting options and to edit file type associations for those options. See **Section 5 - Syntax Preferences Dialog**.

2.1 General Information

Environment settings are the current configuration of the WinIDE environment. This environment information is loaded every time the WinIDE application is run, and saved every time the application exits.

Information stored in the environment includes:

- Whether or not a project is open, and if so, it's name
- Current size, style, and location of the WINIDE main window
- The current Font being used
- The current source directory and project directory
- Options in the Environment Settings, Editor Options, and Preferences dialogs

When the WINIDE environment is run, the environment file (WINIDE.INI) is opened. The first information read from the environment file is whether a project is open, and if so, what the filename of the project file is. If a project is not open, the environment proceeds to read all the environment settings from the environment file. If a project is open, all environment settings and additional desktop information are read from the project file instead. If this way, the user can have the environment configured differently for different projects. For

instance, the user may want to use a different compiler with different projects. The environment 'remembers' all options, including which compiler is used with each project.

2.1.1 Command Line Parameters

The developer can specify the command line options to be passed to each executable program. The name of the currently edited file, or some derivative thereof, can be passed within these options. To pass the current filename, the user should specify a parameter %FILE%. Rapid will substitute this string with the current filename at execution time. The user is also allowed to change the extension of the passed filename, by specifying it within the %FILE% parameter. To specify an .S19 extension on the current filename the user would specify a %FILE.S19% parameter.

For example, If the current filename being edited is COMPDA.ASM:

```
parameters specified parameters passed to program
%FILE% S L DCOMPDA.ASM S L D
%FILE.S19% 1 @2COMPDA.S19 1 @2
```

Although it is by default the currently edited filename which is used in the %FILE% parameter substitution, the environment can be configured to always pass the same filename This is done by checking the MAIN FILE option in the General Environment page. This is useful if the user wants to pass a specific filename to the external program without regard to what's being edited.

3 Environment Settings Dialog

The Environment Settings Dialog contains several tabs that allow the user to manage the IDE environment and aspects of the assembler/compiler and other executables. The user can navigate to the EXE4 tab, not visible in **Figure 3-1**, by clicking the right arrow to move along the tab bar.

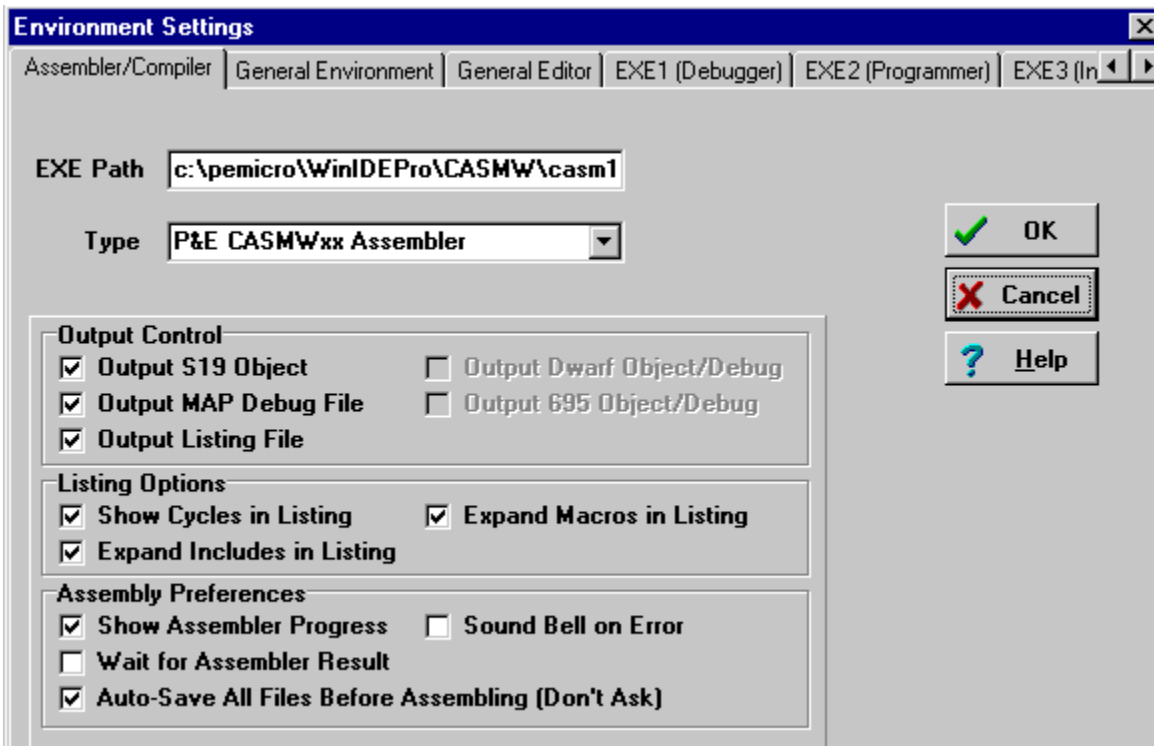


Figure 3-1: Environment Settings - Right Arrow May Be Used To View EXE4 (Simulator) Tab

3.1 Assembler/Compiler

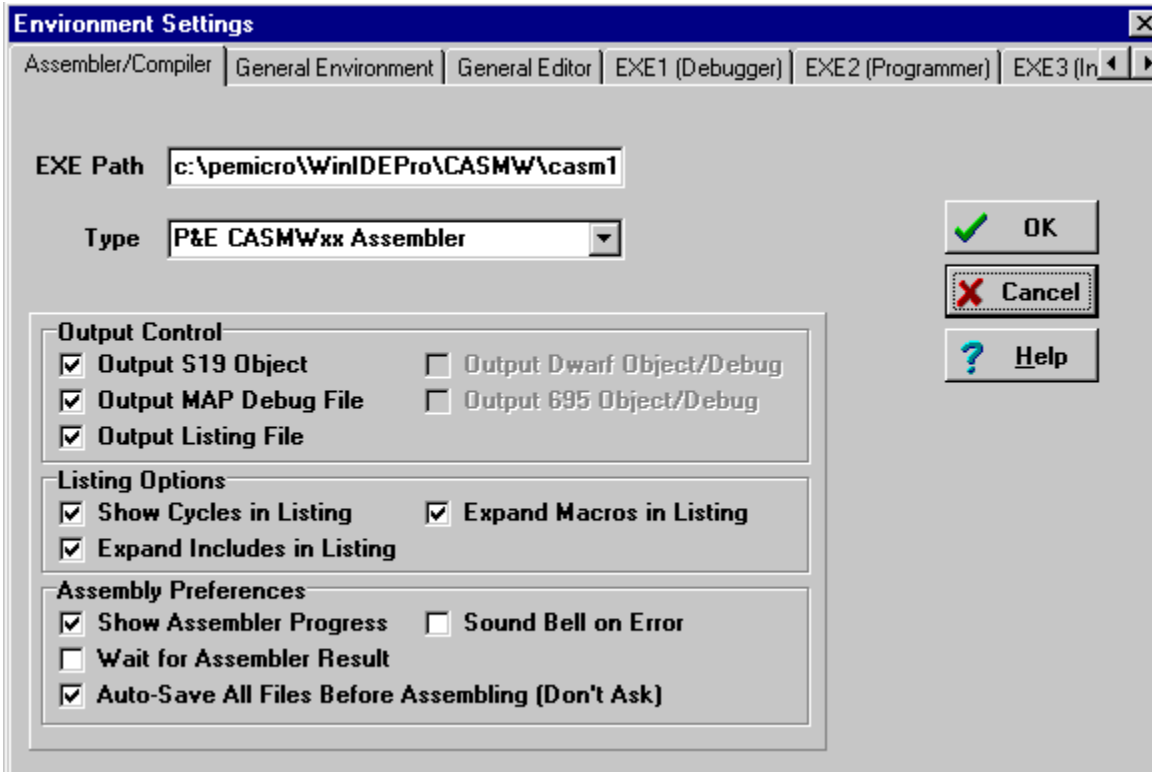


Figure 3-2: Assembler/Compiler Tab

EXE Path

This edit box is where the user should specify the full path and executable name of the compiler. The executable name can have extensions EXE/COM/BAT. For a DOS executable or BATch file, the user may want to create a PIF file so the screen doesn't change video modes when the file is run.

TYPE

The type listbox allow the user to choose what type of assembler they are running. If a PEmicro assembler is selected, a whole panel of checkboxes are displayed for the user to enable the options they want. This is because the editor is 'smart' and knows what parameters to pass the assembler. If the type is not a PEmicro assembler, the user is given several edit boxes to specify the parameters they want to pass the assembler. Following are two sections that describe the options for PEmicro assemblers and NON-PEmicro assemblers.

3.1.1 COMPILER TYPE = P&E

3.1.1.1 Output Control

Output S19 Object

If this option is checked, an S19 object file will be produced by the assembler. This object file holds all the compiled instructions from the program which was assembled. The output S19 file has the same name as the assembly file, but with an .S19 extension.

Output Debug File

If this option is checked, a debug MAP file will be produced by the assembler. This debug file holds symbol information as well as line number information for source level debug from the program which was assembled. The output debug file has the same name as the assembly file, but with a .MAP extension.

Output Listing File

If this option is checked, a listing file will be produced by the assembler. The listing file show all the user's source code as well as the object codes which were produced from the assembler. This file is very useful for the user to see exactly where and how their code was assembled. The output listing file has the same name as the assembly file, but with an .LST extension.

3.1.1.2 Listing Control

The following options specify how the listing file is generated.

Show Cycles in Listing

If this option is checked, the listing file will include cycle information for each compiled instruction. This will allow the user to see how long each instruction will take to execute. The cycle count appears to the right of the address and is enclosed in brackets.

Expand Includes in Listing

If this option is checked, the listing file will expand all include files into the current listing file. Thus the user will see all source files in the one main listing file. If this option is not checked, all the user will see of each included file is the \$Include define.

Expand Macros in Listing

If this option is checked, the listing file will expand all macros into the listing file. Macros are series of instructions that are bundled into a single user-defined mnemonic. If the option is checked, every time the macro is used the instructions comprising the macro will also be show. If the option is not checked, then the user will only see the macro name, and not it's internal instructions.

3.1.1.3 Assembly Preferences

Show Assembler Progress

If this option is checked, a window pops up showing the current assembly status including what pass the assembler is on, what file is currently being assembled, and what line is currently being assembled. If this option is not checked, then the window will not be shown and the user will have to wait for the result to be displayed on the status bar at the bottom of the environment window.

Wait for Assembler Result

If this option is checked and the "Show Assembler Progress" option is checked, the progress window doesn't disappear by itself when assembly is done. The window will stay displayed with the assembly result until the user clicks the OK button. Usually this option is not checked because the result is displayed on the status bar at the bottom of the environment window.

Save Files before Assembling

If this option is checked, all open files are saved to disk before the assembler is run. This is very important because the assembler/compiler reads the file to be compiled from the disk and not from the memory of the environment. If the file being assembled isn't saved, the user will be assembling the last saved version. It is recommended that the user leave this option checked.

Sound Bell on Error

If this option is checked, the assembler will beep if there was an error from the assembler.

3.1.2 Compiler Type = GNU

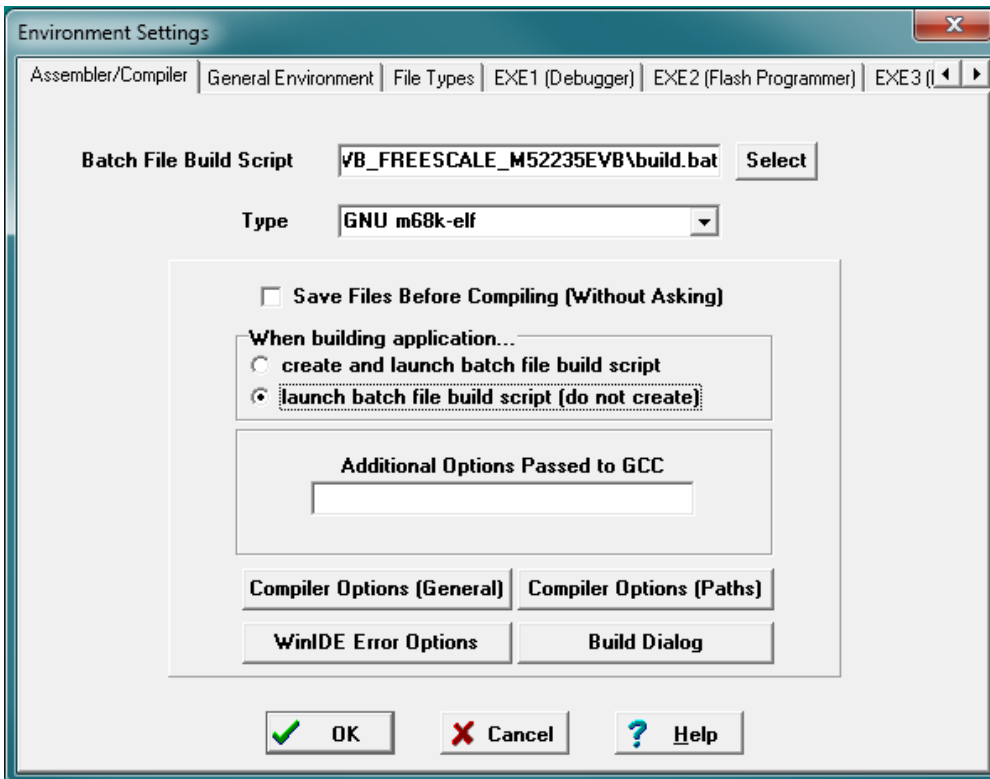


Figure 3-3: GNU Compiler Type Selected

3.1.2.1 Compiler Preferences

Save Files before Compiling (Without Asking)

If this option is checked, all open files are saved to disk before the compiler is run. This is very important because the assembler/compiler reads the file to be compiled from the disk and not from the memory of the environment. If the file being compiled isn't saved, the user will be compiling the last saved version. It is recommended that the user leave this option checked.

3.1.2.2 When Building Application

If the user wishes the batch file to be created before launch, "Create and launch batch file build script" should be selected. If the user wishes to simply launch the existing batch file, "Launch batch file build script (do not create)" should be selected instead.

3.1.2.3 Additional Options Passed to GCC

Additional parameters can be passed to the GCC compiler by entering them in this box.

3.1.2.4 Compiler Options (General)

The options to inhibit warnings, verbosity, optimization levels, and compiler output are located under Compiler Options (General). Please consult the GCC documentation for more information on these options.

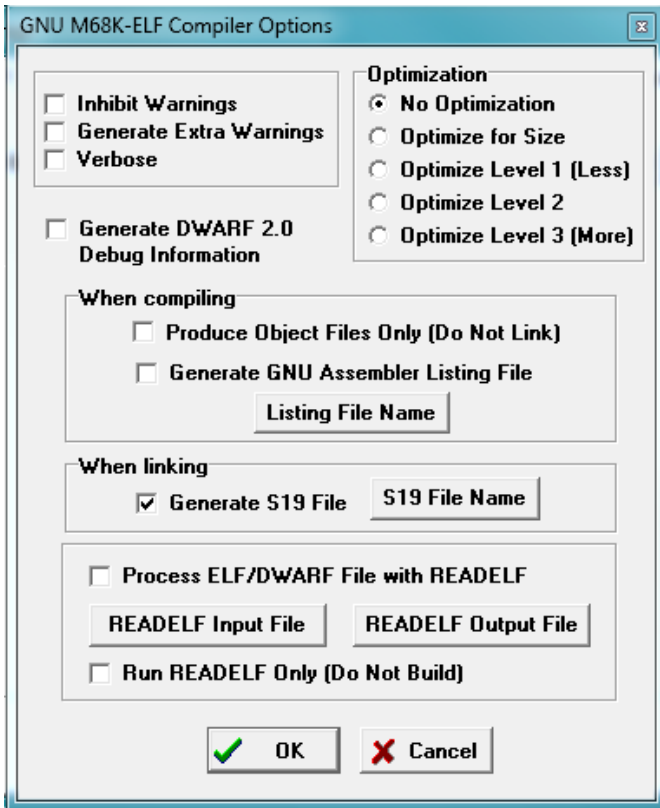


Figure 3-4: Compiler Options (General)

3.1.2.5 Compiler Options (Paths)

The paths of the compiler output, linker script, and search paths can be changed under the Compiler Options (Paths). Please consult the GCC documentation for more information on these options.

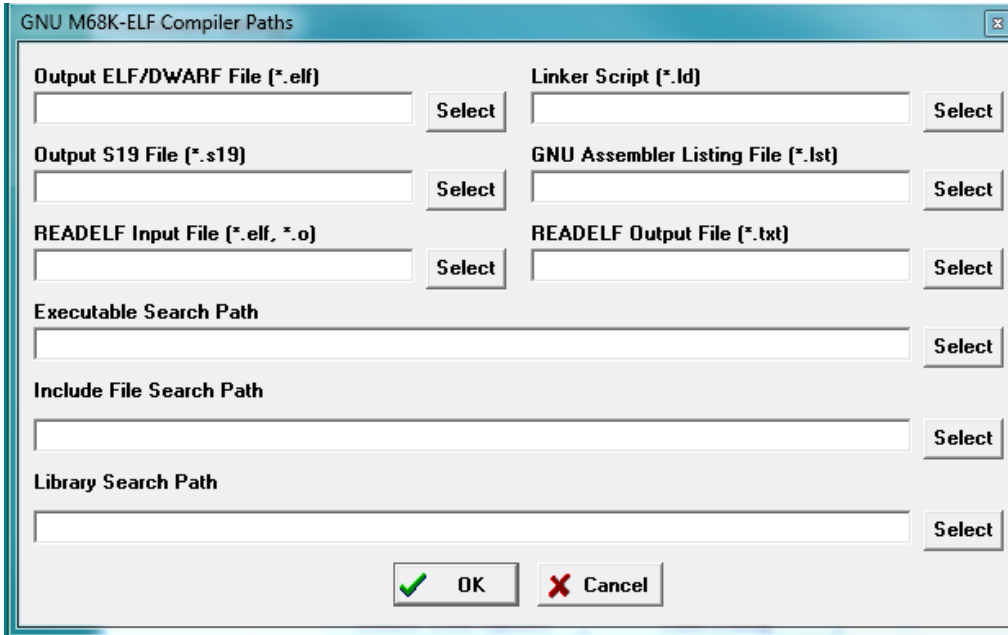


Figure 3-5: Compiler Options (Paths)

3.1.2.6 WinIDE Error Options

The user can use this dialog to set options for error recovery, including the name of the error file and format.

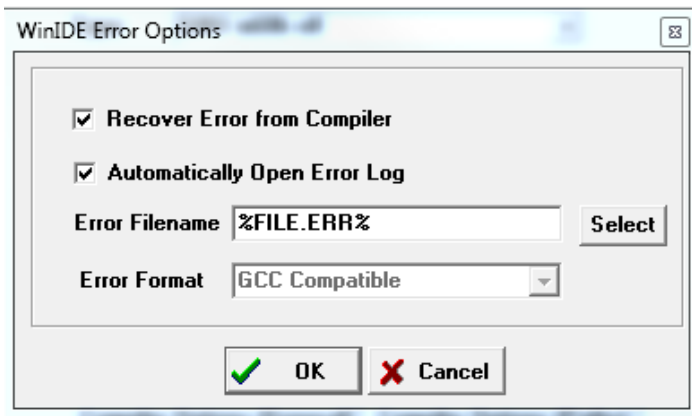


Figure 3-6: WinIDE Error Options

3.1.2.7 Build Dialog

GNU tools will compile and link the files in the order specified under Build. File can be added and removed, and the order changed, using the corresponding buttons.

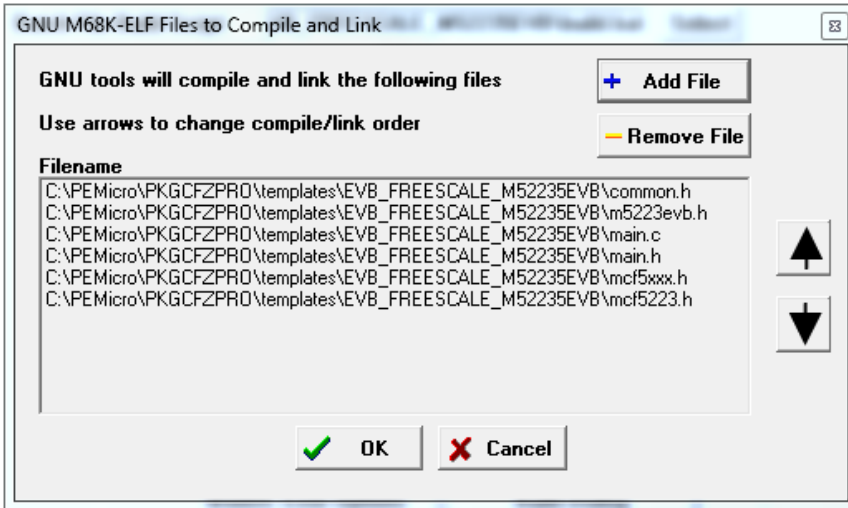


Figure 3-7: Build Dialog

3.1.3 Compiler Type = OTHER

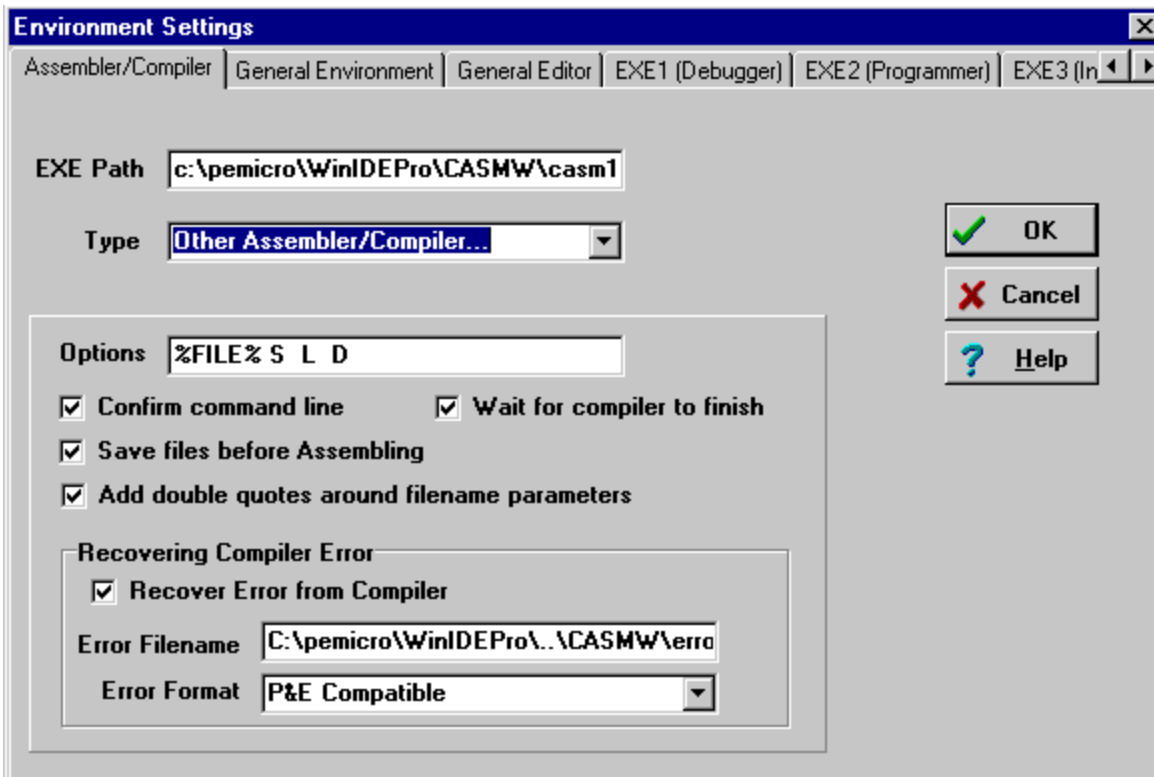


Figure 3-8: Other Assembler/Compiler Selected

3.1.3.1 Options

This input edit box is where the user places the options to be passed to the compiler on the command line.

Most of these options will be switches to tell the compiler what to do, but a filename generally needs to be passed as well. Using the %FILE% string in the command line will insert either the current edit filename or a predetermined filename. For more details, see passing command line parameters.

3.1.3.2 Confirm command line

If this option is checked, just before the assembler/compiler is run, the user will be presented with a window describing the executable about to be run as well as the exact parameters that will be passed. The user has the option to cancel the compiler, continue, or modify the parameters and continue. If the option is not checked, the assembler/compiler will be run without prompting the user for confirmation.

3.1.3.3 Recover Error from Compiler

If this option is checked, the WIN IDE environment will attempt to recover error/success information from the compiler. If an error is detected, the environment will open the file and highlight the error line as well as displaying the error on the status bar. For this feature to work the "Error Filename" and "Error Format" options must be set correctly. If the options is not checked, the environment will not look for a compiler result and will not display the results on the status bar.

3.1.3.4 Wait for compiler to finish

If this option is checked, the WIN IDE environment will disable itself until the compiler terminates. This is necessary if the user wishes to recover an error from the compiler. It also prevents the user from running external programs from the environment (which may need the compilation results). If this option is not checked, the environment starts the compiler running and forgets about it, letting windows' multitasking take care of the program.

3.1.3.5 Save files before Assembling

If this option is checked, all open files are saved to disk before the assembler is run. This is very important because the assembler/compiler reads the file to be compiled from the disk and not from the memory of the environment. If the file being assembled isn't saved, the user will be assembling the last saved version. It is recommended that the user leave this option checked.

3.1.3.6 Add double quotes around filename parameters

If this option is checked, double quotation marks will be placed around the filename when it is passed to an external program so that any spaces in the filename will not be misinterpreted.

3.1.3.7 Error Format

If the environment is to try and read back an error from a compiler, it must understand the error syntax. This option allows the user to select a specific error format from a list of supported formats. If the "Recover Error from compiler" is checked and the filename specified by " Error Filename" is found the environment parsers that file from end to beginning looking for the error. If an error is found, the environment will open the file and highlight the error line as well as displaying the error on the status bar.

3.1.3.8 Error Filename

If the environment is to try and read back an error from a compiler, it must know where to find the error. The user must pipe the output of their compiler to a file. Sometimes this is a switch on the compiler, and sometimes the user must find a way to pip the output themselves. If the compiler is a DOS compiler (which most are), the user can create a BATch file which pipes the output. If the BATch file has a line such as this:

```
COMPILER OPTIONS > ERROR.TXT
```

This will put the compilers output to the ERROR.TXT file. In the case of most C-compilers, the user will have to have a batch file to run the compiler through its different steps (compiling, linking...), so adding a pipe on the

output shouldn't be difficult.

This edit box must contain the filename of the error file if the environment is to be able to read the error back. Once the environment reads this error file, the file is deleted so it would be incorrectly read in the future. If the user wants a copy of the file, they may simply add a line to the batch file to make a copy of it.

3.2 General Environment

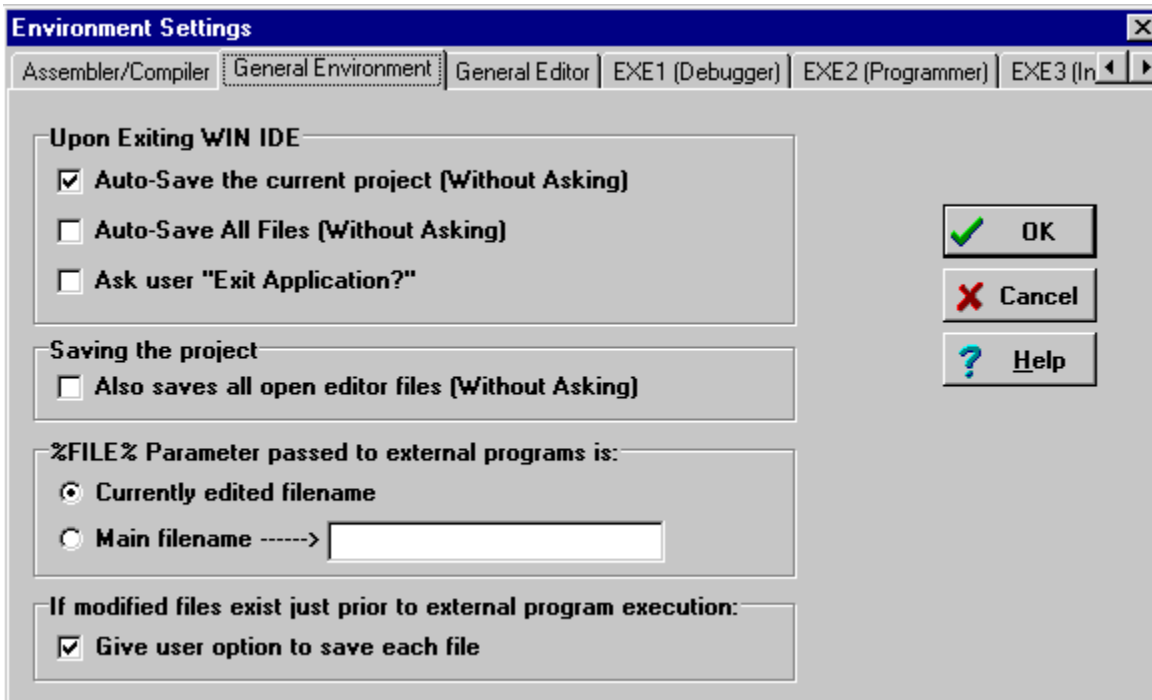


Figure 3-9: General Environment Tab

3.2.1 Upon Exiting WIN IDE...

3.2.1.1 Auto-Save the current project

This setting only has an effect if a project is open when exiting the WIN IDE environment. If this setting is checked, the currently open project will be saved without prompting the user. If WIN IDE is configured to save files when the project is saved, then all currently open files will be saved as well. If this option is not checked, the user will be prompted as to whether to save the open project or not.

3.2.1.2 Auto-Save all Files

If this setting is checked, upon exiting the WIN IDE environment all open editor files will be saved without prompting the user. If this option is not checked, the user will be prompted to save all modified files.

3.2.1.3 Ask user "Exit Application?"

If this option is checked, the user will be prompted to exit the application whenever they try to exit. This allows the user to make sure they really want to exit.

3.2.2 Saving the project...

Also saves all open editor files

If this option is checked, whenever the user saves the current project, all open editor files are saved as well. If this option is not checked, all the project/environment information is written to the project file, but the editor files are not saved.

3.2.3 %FILE% Parameter passed to executable programs is:...

The user is allowed to specify the %FILE% string as a command line parameter for executable programs launched from within the environment. This option specifies what is really passed on the command line in place of the %FILE% string.

Currently edited filename

If this option is selected, whatever edit file is currently in focus has it's filename used in the %FILE% parameter substitution.

Main Filename

If this option is selected, the filename in the MAIN FILE edit box is used in the %FILE% parameter substitution.

3.2.4 If Modified files exist just prior to external program execution:...

All executable programs which can be launched from the WIN IDE environment have the option to save all editor files before the program is launched. If that particular option is not selected for an application, then the situation may arise where an external program is being run while modified files exist in the editing environment. Sometimes this situation is undesirable and may lead to incorrect results.

Give user option to save each file

If this option is checked, the user will have the chance to save each modified file before the external program is launched. Otherwise the external program is run with no warning.

3.3 General Editor

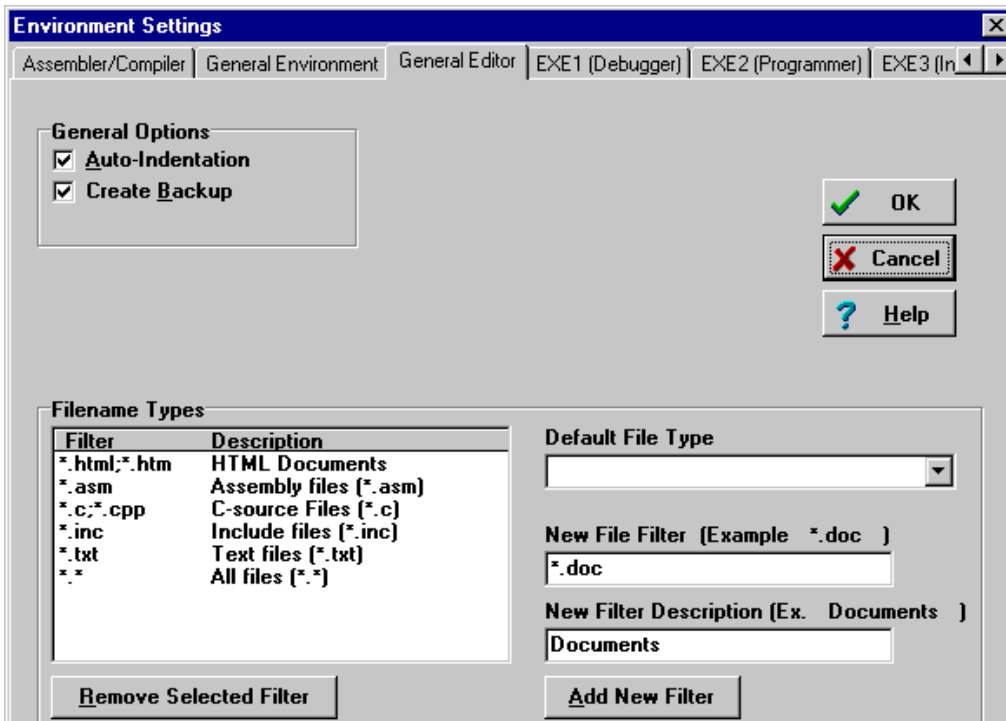


Figure 3-10: General Editor Tab

3.3.1 General Options...

3.3.1.1 Auto-Indentation

The auto-indentation feature determines in what column of the next line the cursor is placed when the user hits the carriage return (enter) key. If the auto-indentation option is checked, the cursor goes to the column of the first non-space character of the previous line. This is extremely useful for writing code as the user doesn't have to space over to always have instructions start in the same column. If this option is not checked, the cursor goes to the first column.

3.3.1.2 Create Backup

If this option is checked, a backup is created whenever a file is saved. Just prior to saving, the IDE will copy the current disk version of the file to the same name with a .BAK extension, and then save the current edited copy over the editing filename. It is highly recommended the user leave this option checked. If this option is not checked, the file will be saved and no backup will be made.

3.3.2 File Filter

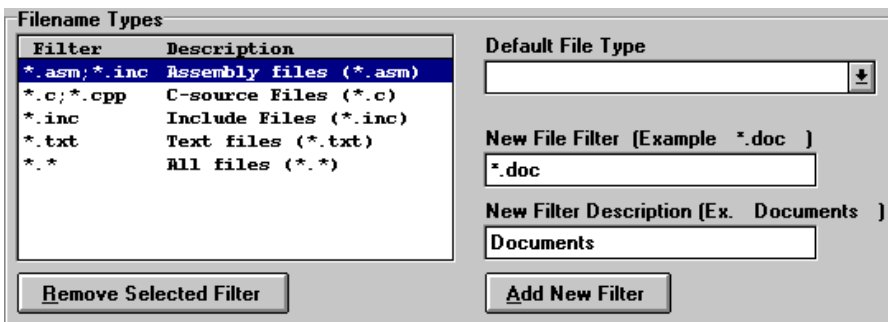


Figure 3-11: File Filter

As part of the Environment Settings / General Editor options, WinIDE allows you to set a default file type by using the file filter, which is useful if you are working primarily with one type of file (although a filter may also include a group of extensions). A list of the filter descriptions and their corresponding extensions is displayed to the left. To select a filter, simply use the Default File Type pull-down box on the right to choose the file type you would like. You can also define your own filters and add them to the list by first entering the appropriate information into the boxes for New File Filter and New Filter Description on the lower right, then selecting the Add New Filter button. Filters that you wish to remove may be deleted by highlighting a filter and using the Remove Selected Filter button.

3.4 Executable 1 (Debugger)

F6 is the hotkey for Executable 1

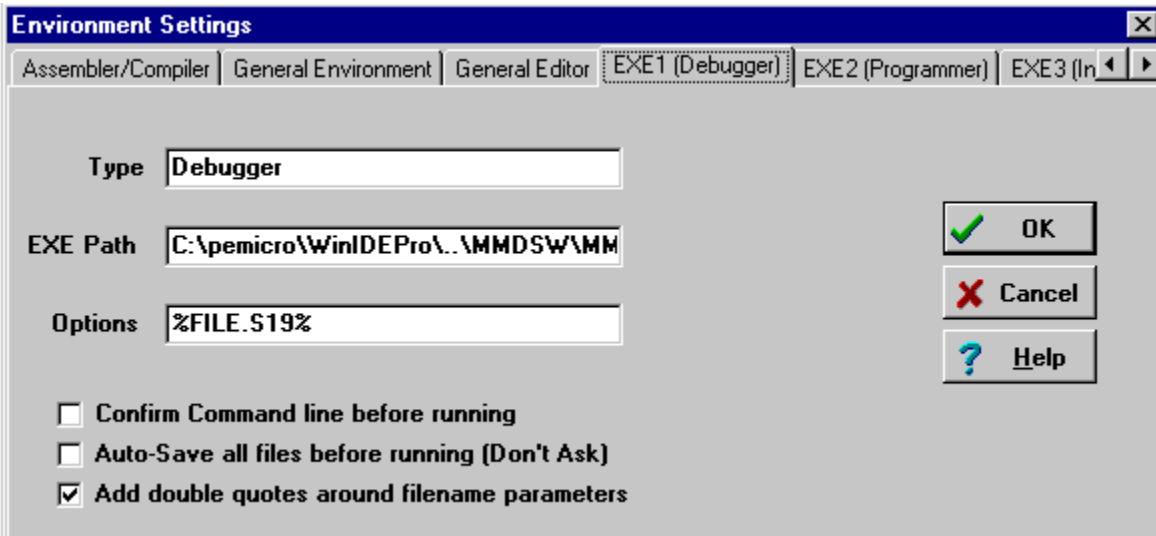


Figure 3-12: Executable 1 Tab

3.4.1 General Page Options

3.4.1.1 Type

The type edit box contains a description of what type of executable is being defined. This string is reflected in other parts of the WinIDE environment. The default for Executable 1 is "Debugger".

3.4.1.2 EXE Path

This edit box is where the user should specify the full path and executable name of the executable program. The executable name can have extensions EXE/COM/BAT. For a DOS executable or BATch file, the user may want to create a PIF file so the screen doesn't change video modes when the file is run.

3.4.1.3 Options

This input edit box is where the user places the options to be passed to the executable on the command line. Most of these options will be switches to tell the executable what to do, but a filename sometimes needs to be passed as well. Using the %FILE% string in the command line will insert either the current edit filename or a pre-determined filename. For more details, see passing command line parameters.

3.4.1.4 Confirm Command line before running

If this option is checked, just before the executable is run, the user will be presented with a window describing the executable about to be run as well as the exact parameters that will be passed. The user has the option to cancel the application launch, continue, or modify the parameters and continue. If the option is not checked, the executable will be run without prompting the user for confirmation.

3.4.1.5 Save all files before running

If this option is checked, all open files are saved to disk before the executable is run. This is very important because some external programs need to read the edit file, and they read only the last version saved to disk. It is recommended that the user leave this option checked.

3.4.1.6 Add double quotes around filename parameters

If this option is checked, double quotation marks will be placed around the filename when it is passed to an

external program so that any spaces in the filename will not be misinterpreted.

3.5 Executable 2 (Programmer)

F7 is the hotkey for Executable 2

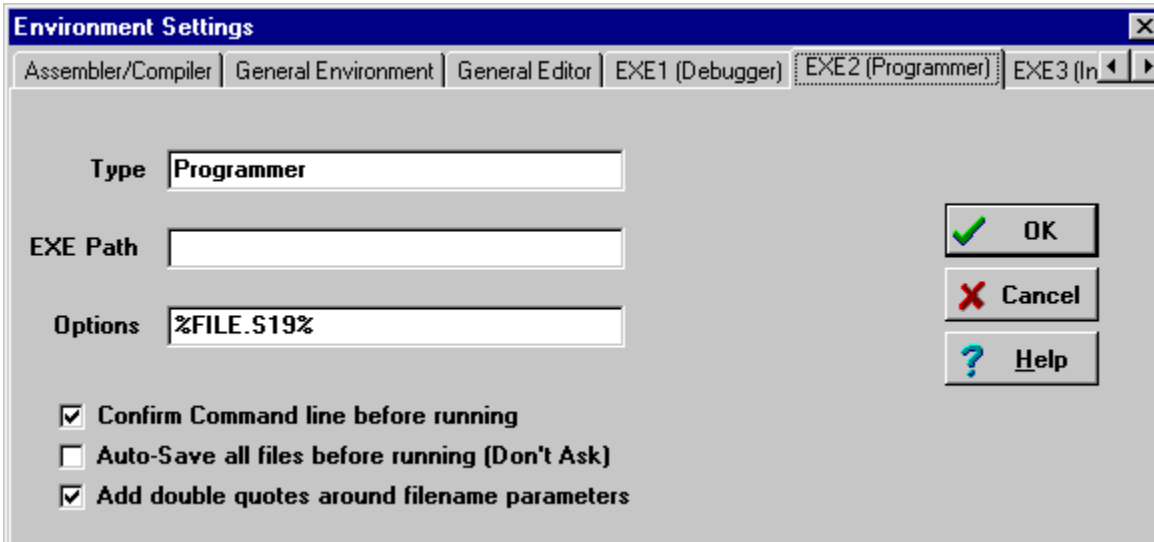


Figure 3-13: Executable 2Tab

3.5.1 General Page Options

3.5.1.1 Type

The type edit box contains a description of what type of executable is being defined. This string is reflected in other parts of the WIN IDE environment. The default for Executable 2 is "Programmer".

3.5.1.2 EXE Path

This edit box is where the user should specify the full path and executable name of the executable program. The executable name can have extensions EXE/COM/BAT. For a DOS executable or BATch file, the user may want to create a PIF file so the screen doesn't change video modes when the file is run.

3.5.1.3 Options

This input edit box is where the user places the options to be passed to the executable on the command line. Most of these options will be switches to tell the executable what to do, but a filename sometimes needs to be passed as well. Using the %FILE% string in the command line will insert either the current edit filename or a pre-determined filename. For more details, see passing command line parameters.

3.5.1.4 Confirm Command line before running

If this option is checked, just before the executable is run, the user will be presented with a window describing the executable about to be run as well as the exact parameters that will be passed. The user has the option to cancel the application launch, continue, or modify the parameters and continue. If the option is not checked, the executable will be run without prompting the user for confirmation.

3.5.1.5 Save all files before running

If this option is checked, all open files are saved to disk before the executable is run. This is very important

because some external programs need to read the edit file, and they read only the last version saved to disk. It is recommended that the user leave this option checked.

3.5.1.6 Add double quotes around filename parameters

If this option is checked, double quotation marks will be placed around the filename when it is passed to an external program so that any spaces in the filename will not be misinterpreted.

3.6 Executable 3 (In-Circuit Simulator)

F8 is the hotkey for Executable 3

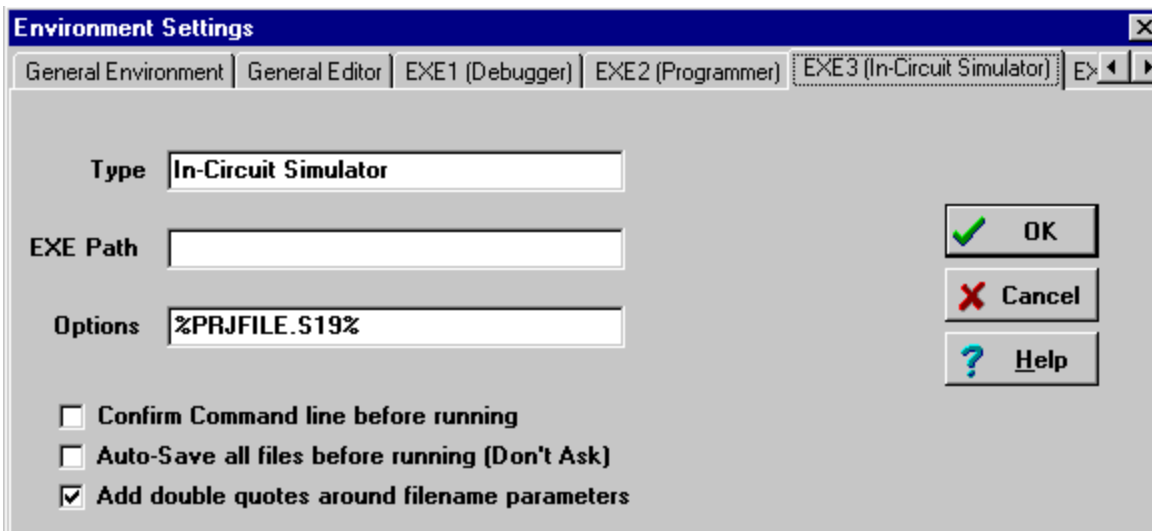


Figure 3-14: Executable 3 Tab

3.6.1 General Page Options

3.6.1.1 Type

The type edit box contains a description of what type of executable is being defined. This string is reflected in other parts of the WIN IDE environment. The default for Executable 3 is "In-Circuit Simulator".

3.6.1.2 EXE Path

This edit box is where the user should specify the full path and executable name of the executable program. The executable name can have extensions EXE/COM/BAT. For a DOS executable or BATch file, the user may want to create a PIF file so the screen doesn't change video modes when the file is run.

3.6.1.3 Options

This input edit box is where the user places the options to be passed to the executable on the command line. Most of these options will be switches to tell the executable what to do, but a filename sometimes needs to be passed as well. Using the %FILE% string in the command line will insert either the current edit filename or a pre-determined filename. For more details, see passing command line parameters.

3.6.1.4 Confirm Command line before running

If this option is checked, just before the executable is run, the user will be presented with a window describing the executable about to be run as well as the exact parameters that will be passed. The user has the option to

cancel the application launch, continue, or modify the parameters and continue. If the option is not checked, the executable will be run without prompting the user for confirmation.

3.6.1.5 Save all files before running

If this option is checked, all open files are saved to disk before the executable is run. This is very important because some external programs need to read the edit file, and they read only the last version saved to disk. It is recommended that the user leave this option checked.

3.6.1.6 Add double quotes around filename parameters

If this option is checked, double quotation marks will be placed around the filename when it is passed to an external program so that any spaces in the filename will not be misinterpreted.

3.7 Executable 4

F9 is the hotkey for Executable 4

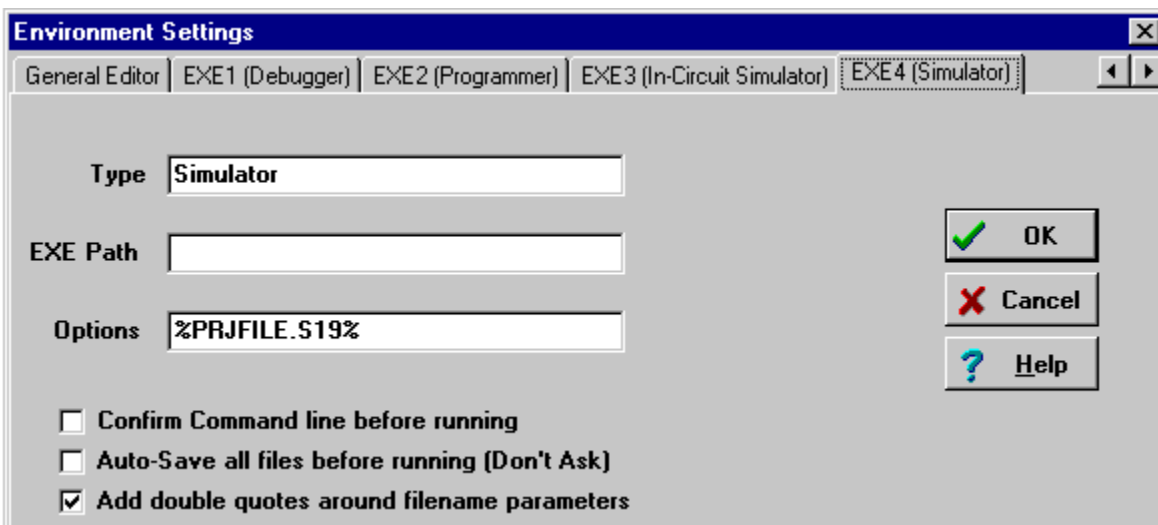


Figure 3-15: Executable 4 Tab

3.7.1 General Page Options

3.7.1.1 Type

The type edit box contains a description of what type of executable is being defined. This string is reflected in other parts of the WIN IDE environment.

3.7.1.2 EXE Path

This edit box is where the user should specify the full path and executable name of the executable program. The executable name can have extensions EXE/COM/BAT. For a DOS executable or BATch file, the user may want to create a PIF file so the screen doesn't change video modes when the file is run.

3.7.1.3 Options

This input edit box is where the user places the options to be passed to the executable on the command line. Most of these options will be switches to tell the executable what to do, but a filename sometimes needs to be passed as well. Using the %FILE% string in the command line will insert either the current edit filename or a pre-determined filename. For more details, see passing command line parameters.

Confirm Command line before running

If this option is checked, just before the executable is run, the user will be presented with a window describing the executable about to be run as well as the exact parameters that will be passed. The user has the option to cancel the application launch, continue, or modify the parameters and continue. If the option is not checked, the executable will be run without prompting the user for confirmation.

3.7.1.4 Save all files before running

If this option is checked, all open files are saved to disk before the executable is run. This is very important because some external programs need to read the edit file, and they read only the last version saved to disk. It is recommended that the user leave this option checked.

3.7.1.5 Add double quotes around filename parameters

If this option is checked, double quotation marks will be placed around the filename when it is passed to an external program so that any spaces in the filename will not be misinterpreted.

4 Editor Options

4.1 Options Tab

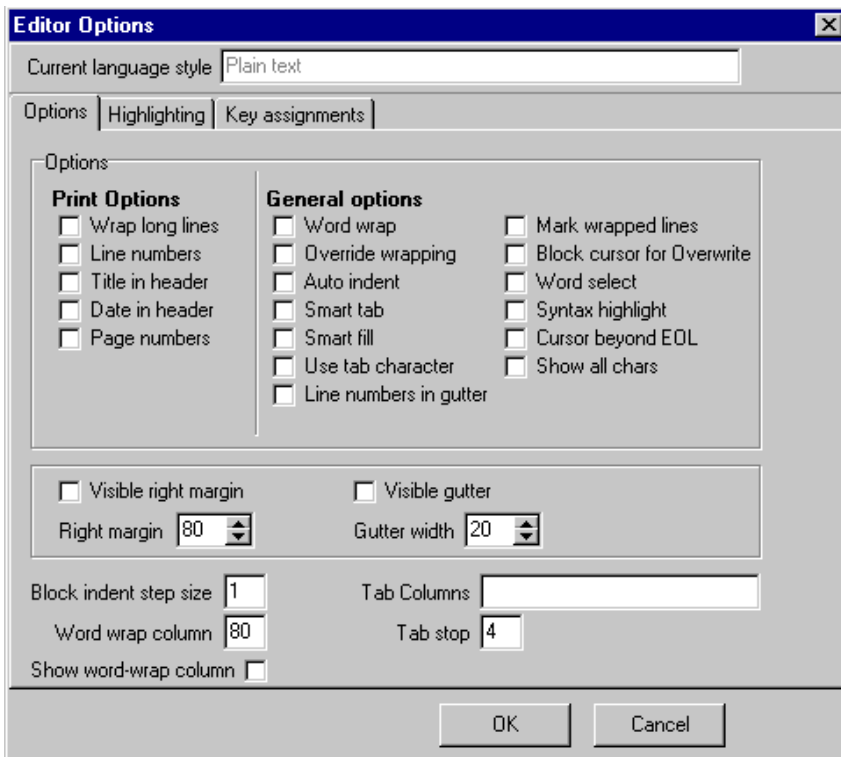


Figure 4-1: Editor Options - Options Tab

4.1.1 {Current Language Style

Displays which language style syntax is associated with the file type (extension) of the currently displayed file.

4.1.2 Print Options

4.1.2.1 Wrap Long Lines

Wraps long lines when printing so that they fit on the page. Overrides word wrap settings (if any) under General Options.

4.1.2.2 Line Numbers

Prints line numbers to the left of each line of text when printing, even if gutter and line numbers are not set to visible under General Options.

4.1.2.3 Title In Header

Prints the filename at the top of the page when printing.

4.1.2.4 Date In Header

Prints the date at the top of the page when printing.

4.1.2.5 Page Numbers

Prints page numbers on each page.

4.1.3 General Options

4.1.3.1 Word Wrap

Activates the word wrap parameter in the Word Wrap Column box. Use the Word Wrap Column box to specify at which column you would like wrapping to occur. You may also adjust the column at which word wrapping occurs by clicking and dragging the dotted line in the editing box that represents the word wrap column, if it is set to visible.

4.1.3.2 Override Wrapping

This function is currently disabled.

#Auto Indent

On carriage return, will automatically indent to the column that is underneath the first character in the previous line.

4.1.3.3 Smart Tab

Smart tabs operate differently depending upon the line above the current line in the editor. If the line above the current one has text, then the tab character advances the cursor to the same column as the beginning of the next character group on the previous line. For Instance, the following line would have the following smart tab stops:

```
BRASStart_ISR; Branches to Interrupt
```

```
^^^ ^           ^ ^
```

Please note that Smart Tab works only with fixed-width fonts. If the previous line in the editor does not have text on it, smart tabs operate the same as fixed tabs.

4.1.3.4 Smart Fill

If Use Tab Character is activated, will use a combination of tabs and spaces, as opposed to all spaces, when filling space during indents and tabs.

4.1.3.5 Use Tab Character

Will use tab character instead of spaces when the Tab key is pressed. Assign tab length using Tab Stop parameter.

4.1.3.6 Line Numbers In Gutter

Places line numbers in the gutter to the left of the edit window. Gutter must set to visible to see line numbers.

4.1.3.7 Mark Wrapped Lines

Will display a special character in the gutter when numbering lines to indicate a line that has been wrapped.

4.1.3.8 Block Cursor For Overwrite

Editor will use block cursor when overwriting with the Insert key.

4.1.3.9 Word Select

Allows you to quickly select an entire word by double-clicking on the word you wish to select.

4.1.3.10 Syntax Highlight

Toggles color-coded text highlighting on/off.

4.1.3.11 Cursor Beyond EOL

Allows you to click to place the cursor at a location beyond the end of lines of text. The cursor will otherwise appear after the last character on a given line.

4.1.3.12 Show All Chars

Will show formatting characters, such as spaces and carriage returns, in addition to the normally displayed text.

4.1.4 Additional Options

4.1.4.1 Right Margin

Allows you to toggle right margin visibility and set the number of columns for the right margin.

4.1.4.2 Gutter

The gutter is an area to the left of the text where line numbering and word-wrap indications may appear. These settings allow you to toggle gutter visibility and set the width of the gutter.

4.1.4.3 Block Indent Step Size

Sets the amount of spaces to indent text when using the Indent button in the menu bar.

4.1.4.4 Word Wrap Column

Sets the column number at which a line will be wrapped.

#Show Word Wrap Column

Toggles a visible line that indicates the column where word wrapping occurs.

4.1.4.5 Tab Columns

Tab Columns enables you to set tab stops at particular columns. You must specify the column numbers, separating each with a space. For example, 5 20 40 80 would place tab stops at those columns. Each tab will move the cursor to the next designated column.

4.1.4.6 Tab Stop

Tab Stop sets the length of the tab character.

4.2 Highlighting Tab

The options contained in the highlighting tab allow you to control the attributes of text highlighting for the current language style, displayed at the top of the Editor Option Box. The current language style reflects the code syntax style that is assigned to the file type of the file that is currently open. These assignments can be edited in the Syntax Highlights box.

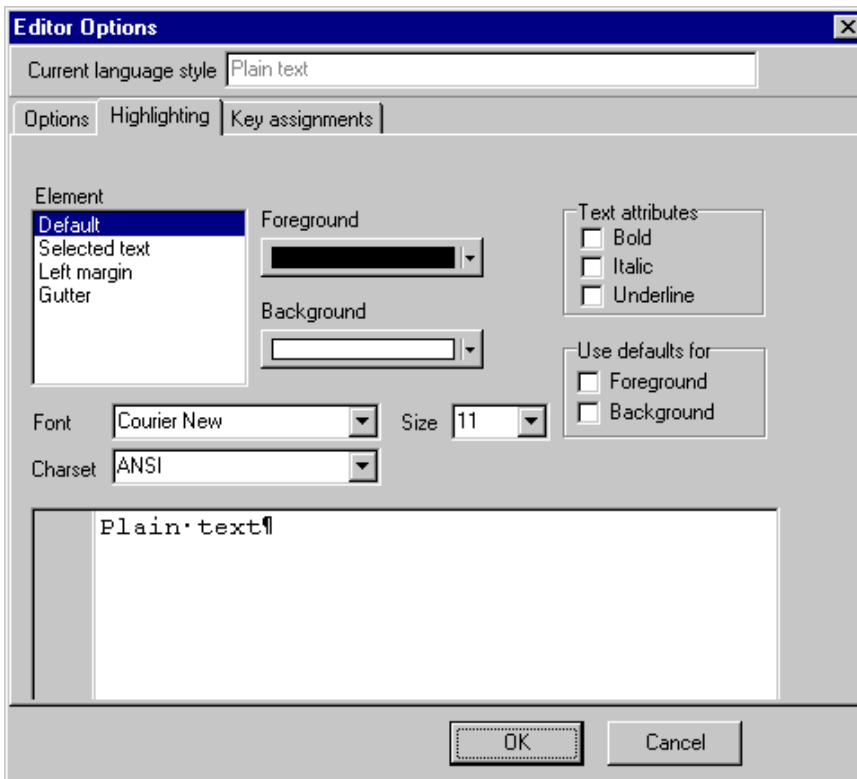


Figure 4-2: Highlighting Tab

4.2.1 Element

The elements associated with the current language style are listed here. Clicking on one of the elements displays the current settings for that element. You may then make modifications to those settings using the options available.

4.2.2 Foreground/Background

This allows you to change the color of the text and the background for whichever element is selected.

4.2.3 Text Attributes

Allows you to add Bold, Italic, and Underline formatting to text.

4.2.4 Use Defaults For

Changes the foreground or background to the default settings.

4.2.5 Font/Charset/Size

Allows you to modify the font and character set and size of the text.

4.2.6 Element Display

Shows a sample of the elements and text formatting associated with the current language style. Will display the assigned color formatting when syntax highlight is selected in the Editor Options box.

4.3 Key Assignments Tab

The key assignments tab allows you to create or modify hotkey combinations for various command functions.

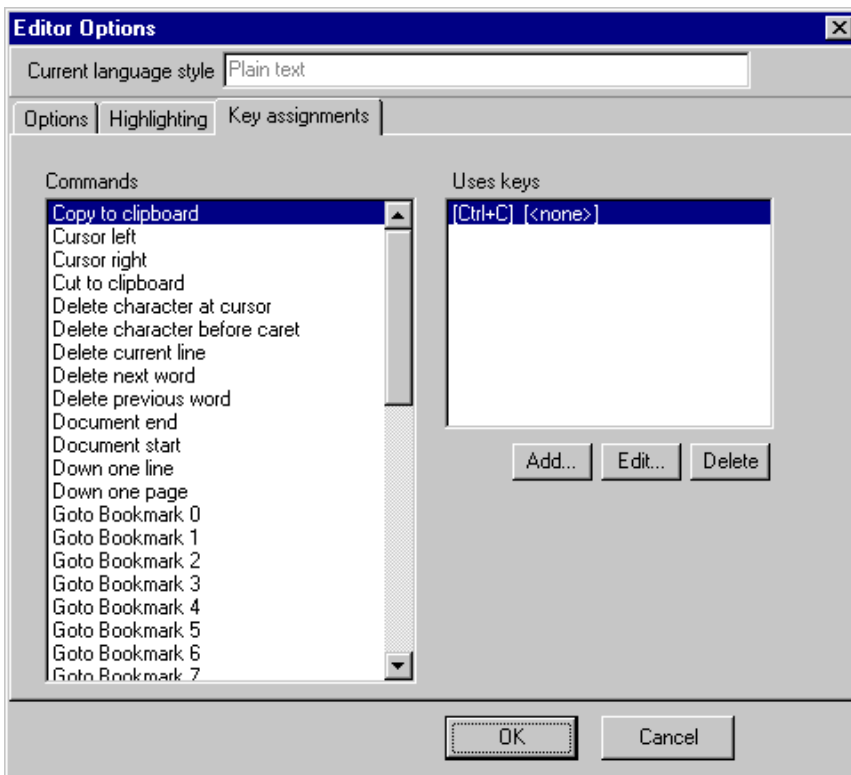


Figure 4-3: Key Assignments Tab

4.3.1 Commands

This is a list of the command functions. Select one in order to modify the hotkeys for that command.

4.3.2 Uses Keys

Displays the hotkey combination for the currently selected command. Use the add, edit, and delete keys to modify the keys associated with particular commands.

5 Syntax Preferences Dialog

Click on an area of the image to display information about that area.

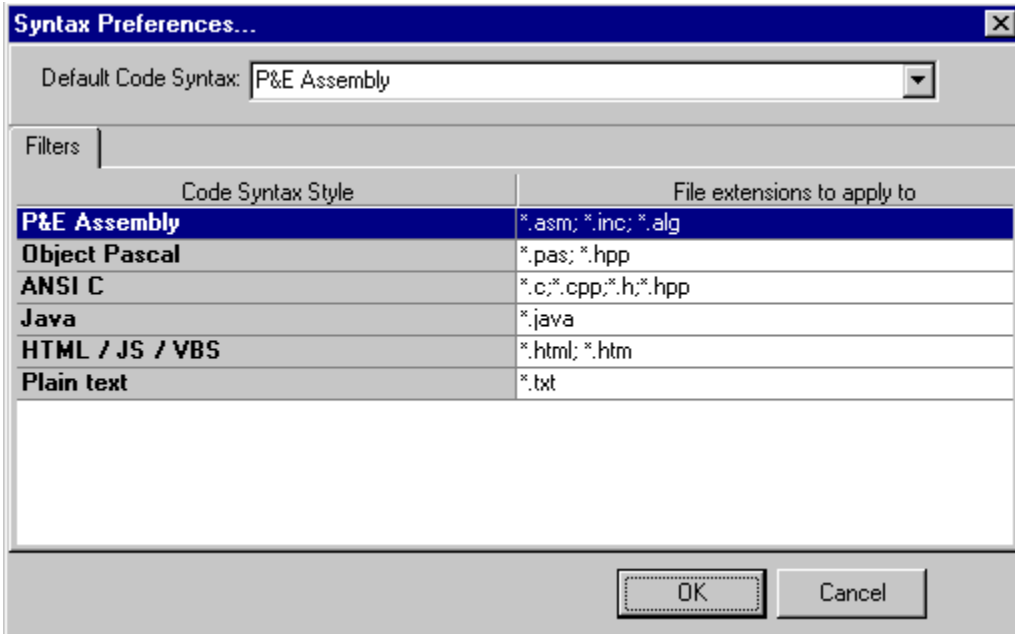


Figure 5-1: Syntax Preferences Dialog

5.1 Default Code Syntax

This is a drop-down list that allows you to choose a Code Syntax. Any files that you open in WinIDE which have a file extension that is associated with that Syntax (as listed in Filters) will use the settings associated with that Syntax Style.

5.2 Filters

Lists the Code Syntax Styles currently supported by WinIDE, along with the extensions that are associated with each Style. You may edit the extensions that are associated with a particular Style by clicking on that Style and making the desired changes in the "File extensions apply to" box.















6 Externally Executable Programs

In addition to running an external compiler, often the developer will want to run other programs such as programmers, debuggers, and simulators. WinIDE allows configuration of up to four external programs, two general purpose programs and one compiler. This configuration of each executable is set up in the Environment Settings page.

7 Main Bar Buttons



Figure 7-1: Menu Bar Buttons

-  **Assemble/Compile Program** - This runs the external assembler/compiler defined in the Environment Settings page under Assembler/Compiler
-  **Executable Program 1** - This button runs the external program defined in the Environment Settings page under Executable 1.
-  **Executable Program 2** - This button runs the external program defined in the Environment Settings page under Executable 2.
-  **Executable Program 3** - This button runs the external program defined in the Environment Settings page under Executable 3.
-  **Executable Program 4** - This button runs the external program defined in the Environment Settings page under Executable 4.
-  **Undo Last Action** – This button will undo the last action performed.
-  **Redo Last Action** - This button will redo the last action that was undone.
-  **Indent Selected Lines** - This button will indent the selected lines one space (to the right) per click.
-  **Unindent Selected Lines** - This button will un-indent the selected lines one space (to the left) per click.
-  **Open A File** - Pops up a file selection window to allow the user to choose a file to open.
-  **Save Current File** - Saves the currently selected edit file to disk, even if it hasn't been modified.
-  **Close File** - This button will close the current file. You will be prompted to save the file if it has been modified.
-  **Open Existing Project** - This button will open an existing project, and place a list of the files contained in that project to the right of the editor.
-  **Add File To Project** - This button will prompt you to select a file to be added to the currently loaded project.



Remove File From Project - This button will bring up a list of the files included in the current project, which you may use to remove the files you choose.



Save Current Project - Saves the currently open project to disk. This optionally saves all open edit files, depending upon the current General Environment settings



Register File Viewer/Editor - Runs REG software, if installed, which allows the user to set modifications to the register files and write them as code to an open file.

8 Project Files

WinIDE makes it easy to manage multiple files by enabling you to save them as projects. When you save a project, WinIDE creates a project file which stores two types of information:

- 1) Environment Configuration User settings and IDE configuration parameters.
- 2) Desktop Information Open edit windows, size and location, markers...

When you are working with a project, the editor will display a list of all the files included in that project in a column to the right. You may access files in this list by simply clicking on the file that you wish to open. There are also buttons on the main bar to allow you to easily add or remove files from your project, save your project, or open a different project.

8.1 Desktop Information

This is all the information stored concerning what files are currently open in the project. Whenever the user saves the project file, the IDE records information about each window open in the desktop. The information includes:

size, position, style (Max/Min/Normal), and what markers are currently set. Whenever the user opens the project, or if the project is open when the IDE starts, these files are all opened with the same settings.

9 Using Markers

Markers allow the user to easily switch between set locations in an edit file. To set a marker anywhere in the file, put the cursor on the line where the marker should be and hit CNTL-SHIFT-N, where N is a value from 0 to 9. If the gutter is set to Visible (Environment -> Setup Editor -> Editor Options -> Options Tab) then a marker will appear in the gutter at the far left of the line. Whenever the user wants the document to shift to that marker location, the user should hit CNTL-N, where N is a value from 0 to 9 (denoting marker 0 to 9). This is useful if the user is often editing two separated portions of a document. Whenever the user saves a project, the markers for all open edit file are saved as well. When the user next opens that project, the markers will still be set.

Examples:

CTRL-SHIFT-3 ; This keystroke sets marker three at the current cursor location

CTRL-3 ; This keystroke moves the cursor and screen to marker three's location

You may also jump to and toggle markers using the right-click Popup Menu. Simply place the cursor over an editing file and click the right mouse button. The menu contains options for Setting a Marker and Jumping to a Marker.

9.1 Popup Menu

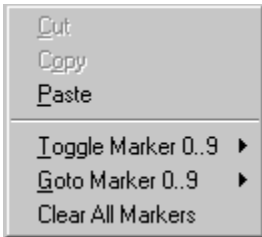


Figure 9-1: Markers - Popup Menu

10 Register Files

PEmicro's register files software enables the Register button on the menu button bar, (or the Shift-F1 command), which opens a processor's register files.

Entering this command opens the register files window, which initially shows a list of register files. Selecting a file brings up a display of values and significance for each bit of the register.

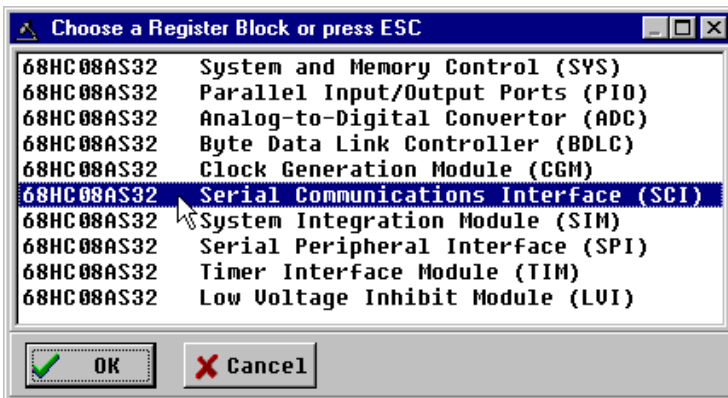


Figure 10-1: Choose Register Block

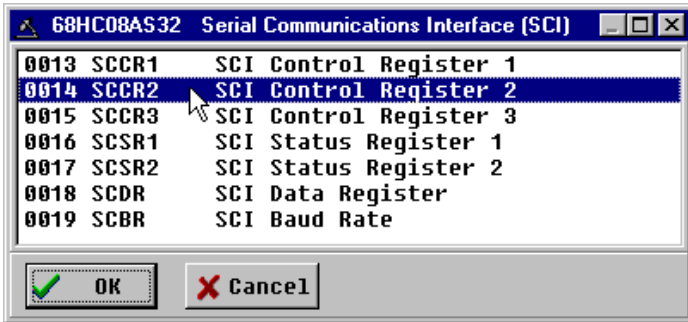


Figure 10-2: Block Register Listing

When the REG software is used from within WinIDE it does not allow real-time modification of the registers - it will not query the board for actual register values or immediately write new values that are input by the user. However, once the user has set a register to a certain value, hitting Enter will write a line of code back to the editor (if a file is open) which contains that register address and new data. This will also set the register once the code is executed.

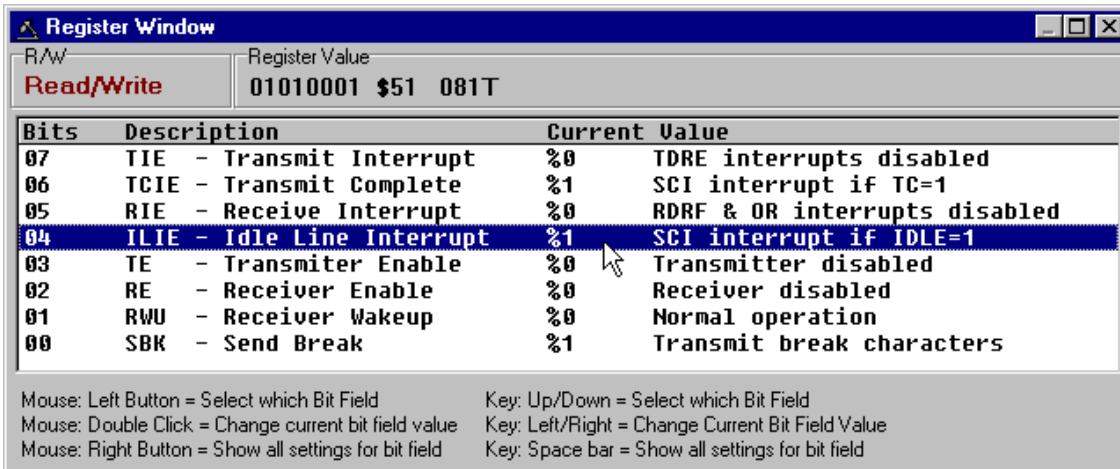


Figure 10-3: Register Window

The user may also use the R command to launch the register window on the command line.

Syntax:

R

Example:

>R Start interactive system register setup.