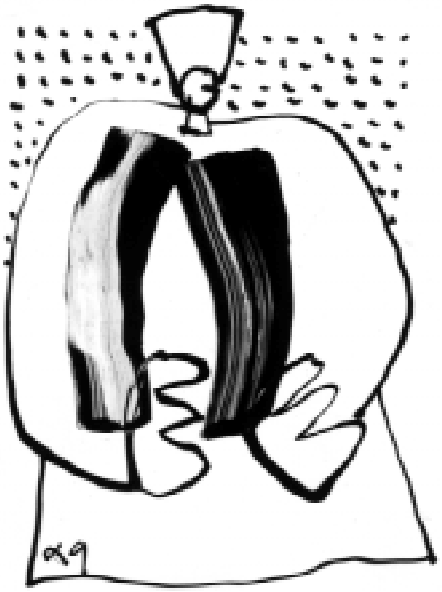
RIDE

Part 1 : Programming



80C51 and 80C51XA
Development Tools

January 2000

RAISONANCE

The information included in this manual may be modified by RAISONANCE S.A and is in no way a commitment of RAISONANCE S.A. The rights of use of the software described in this document may only be transferred under a licensed agreement. Thus, the software may only be used under the terms of this agreement. All copying of this document for any other purpose besides the personal use of the buyer, must have prior authorisation from RAISONANCE SA.
The present manual covers the RIDE software.

REGISTERED TRADEMARKS :
Intel is a registered trademark of INTEL Corporation.
Raisonance is a registered trademark of RAISONANCE S.A.

TABLE OF CONTENTS

1. General presentation

1.1 Organisation and Presentation

1.2 RIDE and software tools

1.1 Organisation and Presentation

This documentation describes the capabilities of the **RIDE** Integrated Development Environment (IDE) and introduces you to application development.

RIDE contains everything you need to write, edit, compile, link and debug your microcontroller application :

- Editor.
- Translation tools : Macro-Assemblers, C Compilers.
- Utility tools : Linker, Library Manager...
- Real Time Kernel.
- Debug tools : Simulator and/or Emulator.

There are several chapters, each devoted to the major elements within the IDE, and for more information on the facilities provided by a particular software tool there is a specific Reference Manual.

The documentation consists of 7 chapters :

1. **General Presentation** (this text)

2. **Installation and invocation**
   Contains information about installing, configuring, and running **RIDE**.

3. **RIDE Interface**
   Provides an overview of the **RIDE** user interface. It describes the basic components of the environment, such as the toolbar, status bar, and Help system.

4. **Project Manager**
   Describes how to create and use projects to build your applications.

5. **Setting Options within a Project**
   How to set the projects' environment and configure compiler, assembler and linker options.

6. **Files, Editing and Windows Management** :
   Discusses how to use the editor to create, open, save, and close files, edit text, move around in files; search for text, and other editing and window management tasks.

7. **Debugging**
   The IDE incorporates sophisticated debugging tools with a simulator for a virtual machine as well as support for in circuit emulators and other development systems. The facilities are discussed in detail in this chapter.

   INDEX

1.2 RIDE and software tools

RKit consists of a number of software tools such as compilers, assemblers and simulators, all of which are Dynamic Link Library files '*.dll'. These *.dll files are called from RIDE.

RIDE can be configured to work with a number of tools but generally the initial configuration is set by the installation process and the serial number provided by RAISONANCE determines the tools that can be fully used and the ones that are available as evaluation versions.

The tools may be supplied separately or as part of predefined 'kits'. The kits available are :

- **RKit51** for the 8-bit 80C51 family, containing the following tools :
  1. ANSI-C RC-51 compiler, with its libraries
  2. MA-51 macro assembler
  3. LX-51 linker
  4. KR-51 kernel
  5. integrated debugger (Simulator and Emulator)

- **RKitXA** for the 16-bit 80C51XA family, containing the following tools :
  1. ANSI-C RC-XA compiler, with its libraries
  2. MA-XA macro assembler
  3. RL-XA linker
  4. KR-XA kernel
  5. integrated debugger (Simulator)

- **RKit51&XA** containing both of the above

2. Installation and invocation

2.1 Installing RKit

2.2 System Requirements

2.3 Installation Procedure

2.4 Invocation of RIDE

2.1 Installing RKit

The « RKit-Installation » program (INSTALL.EXE) installs all the necessary files and correctly configures the environment for running **RIDE**. You can install the kit from CD-ROM or from a network; 3.5 floppy disks are available by special request. The files on the distribution disk are compressed and so the installation program both uncompresses, and copies, them to your hard disk. Before installing **RIDE**, make sure that your computer meets the minimum system and language requirements and close any other running applications. The « README.TXT» file, contains important information about the installation procedure.

2.2 System requirements

RIDE requires the following minimum configuration :

- A 80486 processor (or better) running a 32-bit Microsoft operating system :

Operating System	**Memory Required**
Microsoft Windows 95 or NT	16 megabyte (MB) of RAM

- A hard-disk drive with 30 megabytes of free space. The actual space required depends on the options you select. Additional space is needed for the source-code files and the executable files created from them.

2.3 Installation Procedure

2.3.1 To run the RKit installation program

1 Insert the CD into your CD-ROM drive.
2 The disk should auto-run but if it does not select « *x*:\INSTALL.EXE » (where *x* is the letter representing your CD-ROM drive) from 'Windows Explorer' and press ENTER.
3 Follow the instructions on the screen.

2.3.2 Setup type

You can select :

- **Complete** Setup, to perform a default installation without customisation.
- **Custom** Setup, to select the component you wants to install.

The proposed components for a custom setup are the following :

RIDE interface
This is installed by default.

80C51 Tools
Select this button to install the **80C51 tools** – the compiler and all the related build utilities and libraries. The tools and their utilities are installed in the \BIN sub-directory. The libraries are installed in the \LIB sub-directory. The 'include' files are installed in the \INCLUDE sub-directory. The examples are installed in the \EXAMPLES\C sub-directory.

Emulator PCE 5130C
Select this button to install the **PCE 5130C** interface. This option installs the emulator interface and its related utility files. These are installed in the \BIN sub-directory.

80C51XA Tools
Select this button to install the **80C51-XA** tools and all the related build utilities. The tools and their utilities are installed in the \BIN directory. The libraries are installed in the \LIB directory. The include files are installed in the \INCLUDE directory. The examples are installed in the \EXAMPLES\C sub-directory.

2.3.3 Root Directory

By default, the root directory for the installation is set to 'C:\Ride'. If you want to change this installation directory, type the complete desired path in the **Destination Location** dialog box.

2.3.4 User information

You will find your Serial Number on your CD or on the registration card joined to the documentation. This serial number is necessary for the installation process and must be registered properly. This serial number will define the tools that will run as evaluation versions and the ones that will be completely available. Do not loose this number, as it must be used every time the software is installed.

2.3.5 To uninstall RKit

- First copy all project and source files that you wish to save, then
- Select **Start | Programs | Ride | RKit Uninstall**

2.4 Invocation of RIDE

After you have installed **RKit**, start **RIDE** by selecting **Start | Programs | WREdit32 IDE**. This program gives you access to the Project Manager where you can create, build and debug new applications. (Should you wish to invoke the IDE from DOS, the full command is RIDE.EXE

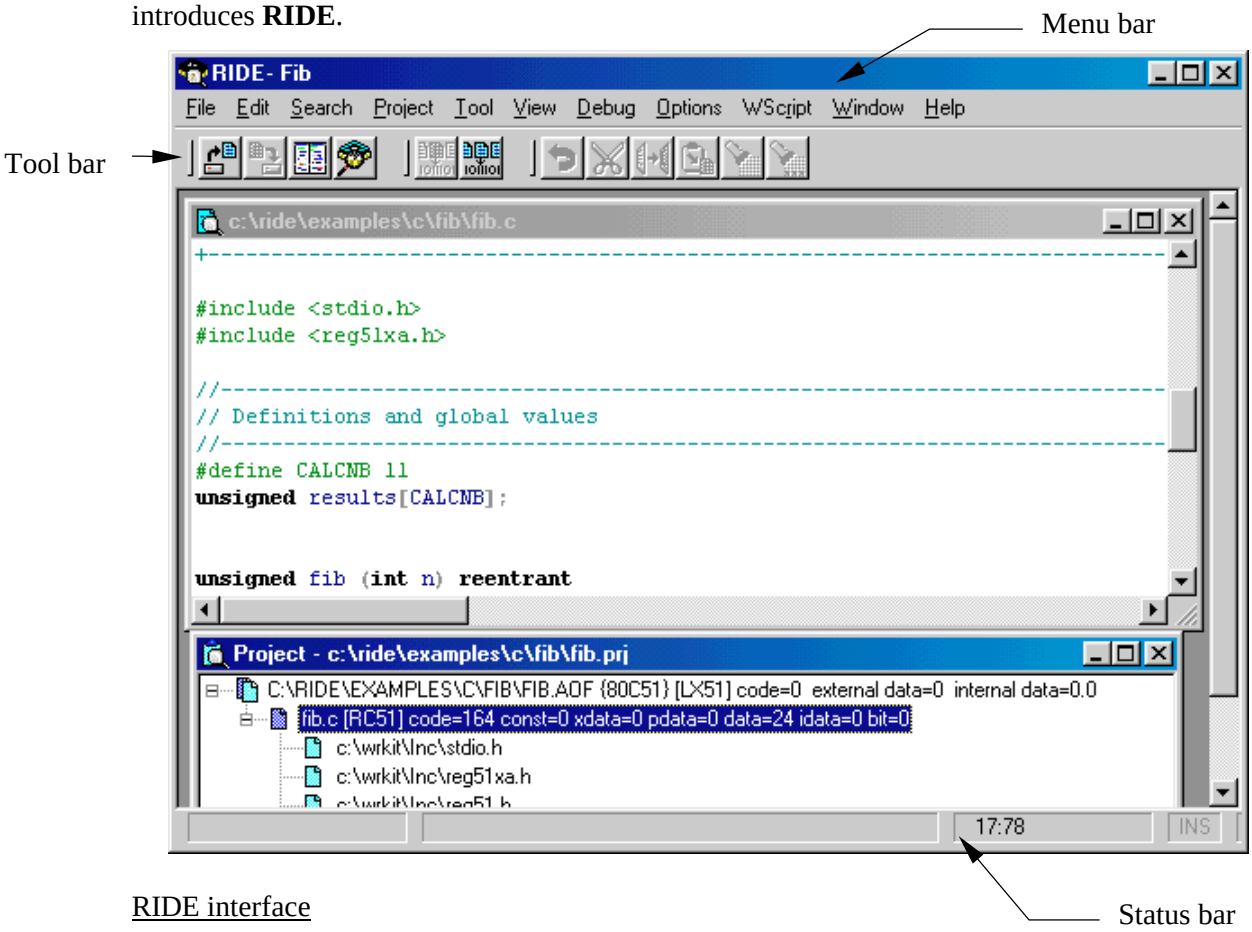
Selecting **Start | Programs | RIDE-Reference** provides access **to** On-line Help for **RIDE**.

If you had a project open when you last exited **RIDE**, it is automatically re-loaded. To quit **RIDE**, choose **File | Exit**.

3. RIDE interface

3.1 Interface Basics

RIDE is an Integrated Development Environment consisting of a text editor, project manager, programming tools' manager, code translation, debugger, and Help database. This chapter briefly introduces **RIDE**.



RIDE interface

3.2 Help

Several types of online help are available.

- **Hierarchical** : select **Help | Index** from the main screen to activate a hierarchical index to various sections.
- **Indexed** : Select **Help | Using help** to engage a standard Microsoft Windows help facility with keyword index.
- **Context sensitive** : With any menu option highlighted press the F1 key for context sensitive help.
- **Help Buttons** : From various dialogue boxes select the help button.
- **Help on error message** : To display more information about an error which occurred during compilation, select the error item in the Message window and then press F1 to display the corresponding topic.

3.3 The text editor

RIDE provides an integrated project manager, which includes a syntax-highlighting text editor to manage, edit, and print source files. Most of the editor's procedures like file and text handling, and moving around in a file, are similar to other Windows-based text editors.

With the editor, you can :

- Perform advanced find and replace operations.
- Perform bracket matching.
- Specify syntax colouring.
- Customise tab size in source files.
- Use toolbar shortcuts for various commands.
- Get context-sensitive Help from within a source file.
- Use multiple levels of undo.
- Open multiple windows for debugging, displaying source files, and viewing project structure.

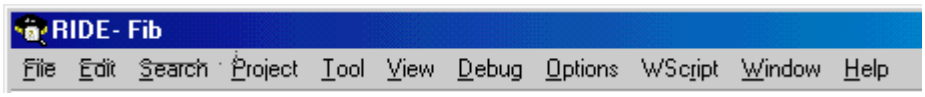
3.4 The Menu Commands

The menu bar provides the primary access to the commands. Each menu consists of a list of items that can be viewed and chosen. When the action represented by a command is not appropriate, the menu item appears in grey text and does not respond to mouse clicks or keyboard selection. Usually the mouse is used to make selections from the menus but the keyboard may also be used and is sometimes quicker for the experienced user.

To display a menu using the keyboard, you can :

- Press the underlined 'hot' key in the menu name or sub-menu while holding down the 'Alt' key.
- If you are currently in the menu bar, and menu pad is already highlighted, you can also press the Left and Right Arrow keys until the menu pad is selected, then press the Down Arrow key to display the menu. Pressing Enter will activate the highlighted command.

To get help about a menu command, you can highlight the command and then press F1 to display the corresponding help topic.



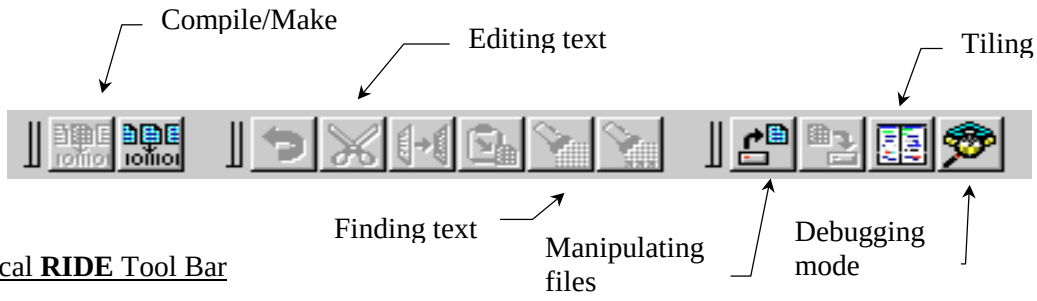
RIDE Menu Bar

3.5 The Tool Bar

The toolbar is a row of buttons, placed beneath the menu bar, which activate commands. Pressing a toolbar button has the same action as choosing the corresponding function from a menu.

Buttons on the toolbar are grouped according to function and may be customised by selecting **Window | Set control buttons**

Typical Buttons Include :

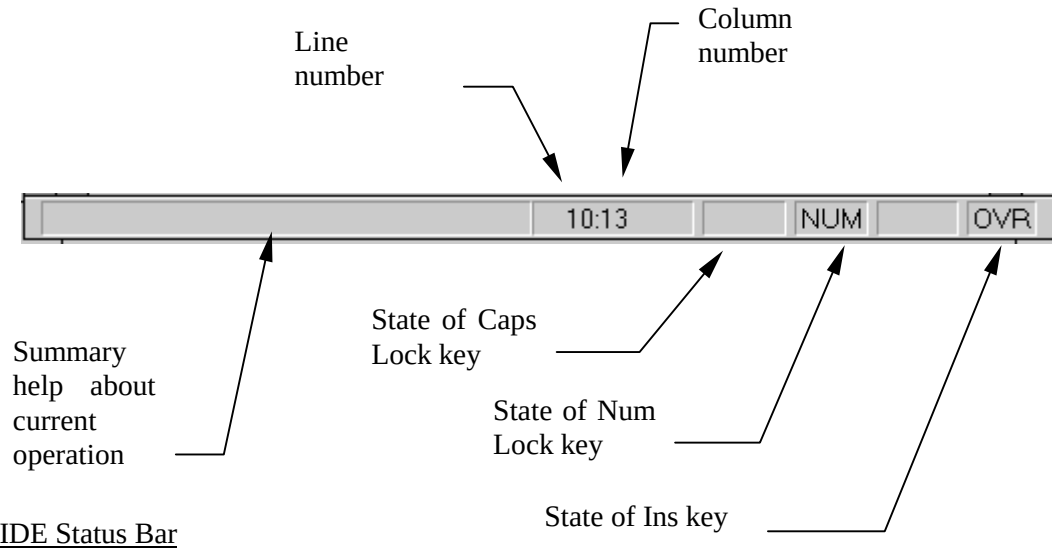


Typical **RIDE** Tool Bar

3.6 The Status Bar

The status bar is a row of "status indicators" in text form. The status bar appears at the bottom of the main window of **RIDE**.

The status bar shows the following information about the active window :



RIDE Status Bar

4. Project manager

4.1 What is a Project ?

4.2 Project Menu

4.3 Tools Menu

4.4 Listing Views

4.1 What is a Project?

The Project Manager helps you to manipulate and work with complex applications. A project contains information about a particular program, the files it uses, the environment, its programming tools and their options. Information is held in two files: a file containing the project options (*.prj) and a file containing the debug options (*.wmc).

RIDE retains information about the project you were working with during your last session. It will display the same desktop of this project at the start of the new session. The project file contains three kinds of information :

- The names and locations of the source files and their dependencies that are used to build your program.
- The settings for the tools required to build the program, such as compiler, assembler, linker or kernel options.
- The organisation of the **RIDE** desktop on your display.

4.1.1 *.PRC and *.PRJ files

The previous WEdit16 (Windows 16-bit) stored information about projects in *.prc files in a different format than *.prj files. However, **RIDE** can open and use projects from WEdit16. Use the **Project | Open** command to open the old *.prc file. **RIDE** converts the old project file to a new one and when closed, this new project is saved with the old name and the new *.prj extension. This facility is provided to allow users of WEdit16 to upgrade easily to **RIDE**.

4.1.2 Project Types

In **RIDE**, two main types of projects are available: 80C51 projects and XA projects. You can select the project type when you first create the project, using the **Project | New** dialog box. Selecting a type of project causes **RIDE** to select the appropriate compiler, assembler and linker.

For a 80C51 project the default building tools are :

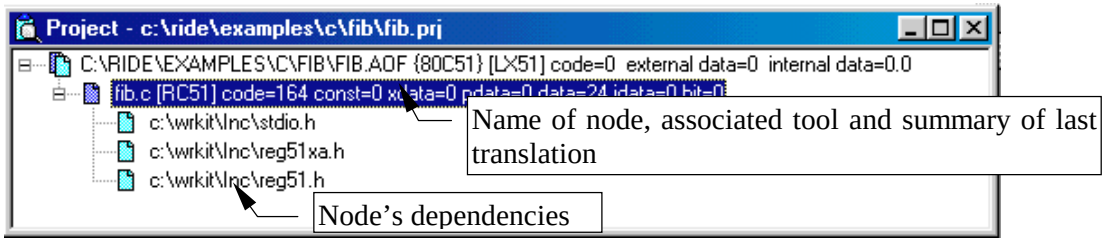
1. RC-51 C Compiler
2. MA-51 Assembler
3. LX-51 Linker

For a 80C51XA project the default building tools are :

1. RC-XA ‘C’ Compiler
2. MA-XA Assembler
3. RL-XA Linker

4.1.3 The project window

The project window displays the different files, their paths and associated tools, which constitute the program/application. If you close this window, it means closing the project. So you can set options, add or delete files from a project only if this window is displayed. The project has a hierarchical structure that may be navigated in a manner similar to Windows Explorer.



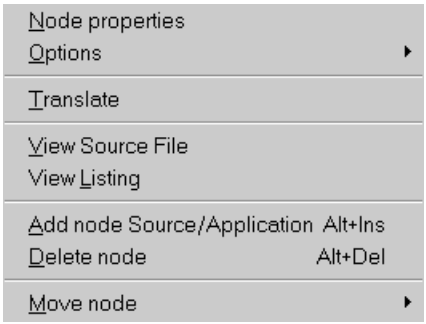
RIDE’s Project View

Within **RIDE**, the project's application is represented as the root of a tree. Each file belonging to it is represented as a node on the tree. Indented below each node are the node's dependencies used to build the node.

Next to the application name, the type of the project (80C51 or XA) and the result of the last successful link (code size, internal data size and external data size) are listed. Adjacent to the application nodes, the build translator, and the result of the last successful compilation or assembly are listed.

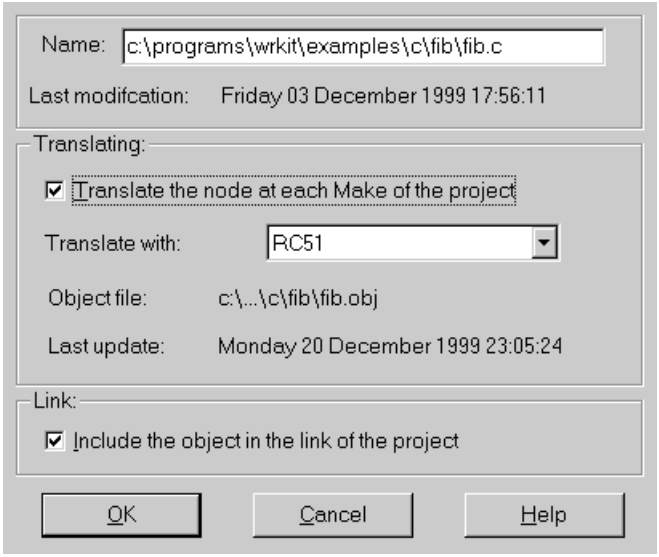
The project options are inherited by the project nodes. However, you can override these options by local options on specific nodes. When a new node is added to the project, a build translator is automatically associated with it according to its file extension. You can select another build translator using the pop-up menu commands.

Pressing the right mouse button within the project window with a node (other than the root) highlighted, causes a pop-up menu to appear :



Non-root node pop-up menu

- **Node Properties:** shows the window below

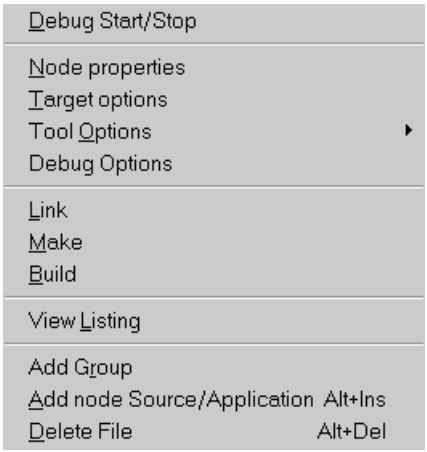


Node Properties window

With this window, the translation and linking of the node may be controlled by checking the appropriate boxes. In addition, the tool used for the translation may be selected. The list of available tools may be modified via the **Options | Tools** command from the main menu.

- **Options | Local Options** : permits local options to be set for the selected node. If local options are used the node in the project view is marked with .
- **Options | Global Options** : selects the global project options when building the currently selected node. Global options are set via **Options | Project** and are described in more detail in the next chapter.
- **Translate** : builds the selected node with its associated build translator.
- **View Source File** : displays the content of the selected node. You can also view the content of a node by selecting the node and pressing ENTER, or merely by double-clicking on the node.
- **View Listing** : displays the content of the listing file resulting from the last compilation or assembly of the node.
- **Add node Source/Application** : With this command you can insert a new node in the project's list. The file filters proposed for selecting files in the **Add File** dialogue box correspond to the programming tools associated with the project.
- **Delete Node** : removes the selected node reference from the project's list. If you want **RIDE** to prompt you before removing the file check the corresponding option in **Options | Project | Environment | Editor**.
- **Move node:** allows the selected node to be moved up (Shift- UpArrow) or down (Shift DownArrow) the list.

When the root node (*.aof) is highlighted and the right mouse button is pressed a pop-up menu appears which is similar to that for other nodes. However, the commands relate to overall management of the project rather than management of an individual node. Although the pop-up menu provides a rapid means to select commands, they may also be engaged from other main menus such as **Project** or **View** and are described in more detail later.



Root node – pop-up menu

Note that the **view | listing** file is actually the output from the linker that shows the memory locations associated with all the nodes and the value of all symbols.

4.2 Project Menu

The main menu contains a sub-menu entitled **Project**, which provides access to 3 groups of commands :

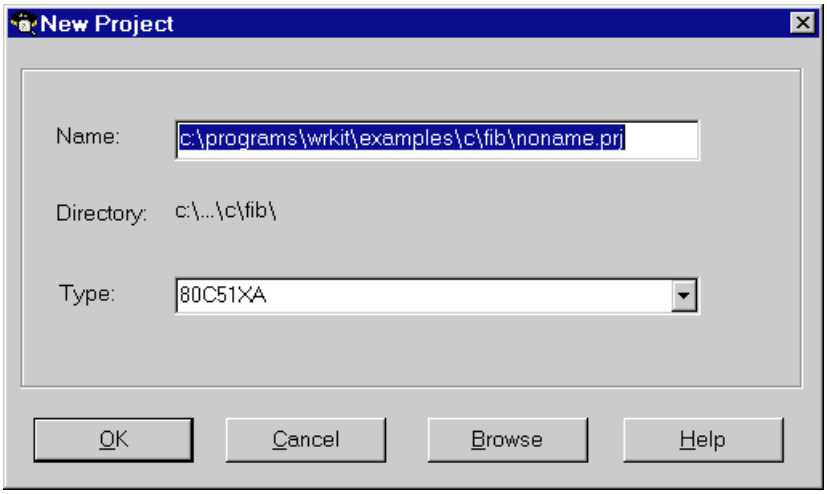
- **New**, **Open**, **Close** and **Save**
- **New Application**, **Add node source/application** and **Delete node**
- **Translate**, **Link**, **Make all** and **Build all**.

Once a project is open and the project window is visible, pop-up menus are available as described in the previous section.

4.2.1 Creating a new project

The first step in developing a program/application with **RIDE** is to create a project. To do this :

1. From the **Project** menu, choose **New**. The **New Project** dialog box appears.
2. Click the **Browse** button to display the **Open New Project** dialog box.
3. Choose the directory and the name of your project file and select **Open**.
4. Choose the type of processor, either 80C51 or 80C51XA.
5. Choose **OK** and a project window will appear.



New Project Dialog Box

The type of processor/project will determine the default building tools and configure the project's environment according to these tools. For a particular application the default options may be inappropriate and the user may select specific options such as the memory model used by the compiler via **Options | Project** from the main screen.

If a project exists already, **RIDE** prompts you to confirm that you want to overwrite the old project file. When the project is created, **RIDE** changes the working directory to the project directory and displays the Project window, so that you can add source files to your project and set its different options.

4.2.2 Opening an existing project

To open an existing project :

1. Choose **Project | Open** and the **Open** dialog box appears.
2. Choose the directory and the name of the project you want to open and select **Open**, or double–click on the filename. **RIDE** will display the project window and all associated windows already opened during the last session. Previous options will also be restored.

4.2.3 To add nodes/files to a project

1. Choose **Project | Add node Source/Application** and the **Add File** dialog box appears.
2. You can also choose **Add node Source/Application** in the pop-up menu, which is opened by clicking the right mouse button when a node is highlighted in the project window.
3. Select the file type to display from the **List Files of Type** drop-down list box. The files of the chosen type(s) appear in the **File Name** list box. The extensions listed can modified by changing the default 'C' source files and assembler files via **Options | Project**. Several files can be selected in the same time.

4.2.4 To remove a node/file from a project

1. Highlight the node you wish to remove using the left mouse.
2. Then select **Project | Delete node**. Alternatively, choose **Delete node** from the pop-up menu, which is displayed by clicking on the right mouse button.

4.2.5 Translate

When an edit window is active, selecting Project | Translate will translate the file according to its type. When an edit window is not active this option appears in grey and may not be selected.

4.2.6 Link

This command engages the linker but not the translators. It is usually used when the linkers options have been changed but not the source files.

4.2.7 Make all

Any source file associated with the project is translated if its time and date stamp is not the same as the corresponding output file. As a project is developed it is usual for only one file to be modified at a time and using the **Make all** command results in reduced translation time as only the files which have been changed are re-translated. In effect, the **Make all** command rebuilds only the files that are not current.

The compilers and assemblers generate two files :

1. A listing file, which contains information relating to the compilation, and includes a list of all detected errors.
2. The object file containing the generated code in OMF-51

These files have the same name as the source file, but with *.obj extension for object files and *.lst for the listing files. The output directory is set via the **Options | Project** dialog box.

A Compile Status information box displays the results of the compilation. If any errors occurred a Message Window becomes active and highlights the first error or warning.

4.2.8 Build all

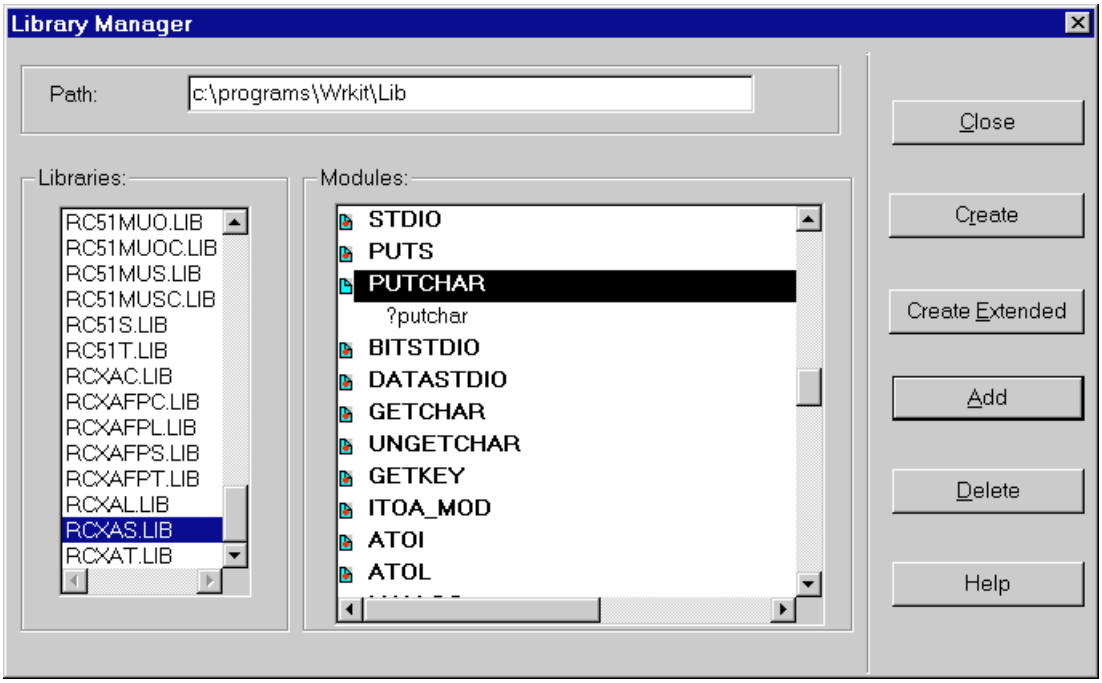
This command rebuilds all files using the tools associated with each file type. This is usually inefficient as files that have not changed will be re-translated but the command is useful to ensure that all object files match their corresponding source files. In effect, the **Build all** command behaves like the make all command where all source files are assumed to have been changed.

The tool associated with a source file depends on its extension. By default 'C' source files have a *.c filename and are processed by a 'C' compiler, while assembly files with either *.a51 or *.axa filenames are processed by a macro assembler. If a node references a file with a non-standard extension, you must associate it with a desired processing tool. Select **Options | Tools | Help** for more information on how to associate a particular tool with particular file-type.

4.3 Tool Menu

4.3.1 Library manager

Select **Tool | Library manager** to show the Library Manager dialogue box. This lets you modify library files without leaving **RIDE**.



Library Manager dialogue box

- **Path** : Type the path of your library directory. The list of the different library files will be displayed in the Libraries list box.
- **Libraries** : This list box lets you select the library file you want to modify.
- **Modules** : This list box displays the different modules defined in the currently selected library. Click the bitmap to the left of the module's name to show (eg. ?putchar) or hide the list of the associated Public Symbols.
- **Create** : To create a new library. Pressing this button displays a dialog box to let you choose the name and location of the new library.
- **Create Extended** : to create a new library for the XA family. Pressing this button displays a dialog box to let you choose the name and location of the new library.
- **Add**: Press the Add button to add a new module to the library currently selected in the Libraries list box. A dialogue box lets you choose the name and location of the new object file to insert into the library.
- **Delete**: Press the Delete button to remove the module currently selected in the Modules list box from the library.

4.3.2 Grep – Global Regular Expression Print

Sometimes it is very useful to find the location(s) of a particular string in the files associated with a project. **Tool | Grep** uses a regular expression for search matches in files with a given extension.

4.3.3 Running a tool

With the **Tool | Run tool** command, you can run a DOS application without leaving the **RIDE** environment. The command shows a list of tools where you can select the one you want to execute. To add or delete an application from this list use the **Options | Tools** command.

4.4 Listing views

4.4.1 Map report from linker

Depending on the type of project/processor that has been selected different linkers will produce different map files. For the 80C51, tool LX-51 produces a *.m51 file, while for the 80C51XA, tool RL-XA produces a *.mxa file. **View | Map Report from linker** displays the appropriate file and thereby provides information about the link process. The format is :

1. The invocation command line.
2. Object modules included in the project.
3. The memory map.
4. The overlay map.
5. The symbol table of public, local, and line number information.
6. A cross-reference table.

4.4.2 Listing from Compiler

When a source file associated with a particular node is opened for editing **View | Listing from compiler** may be used to display the most recent associated list file (*.lst). The same file may be displayed via the pop-up menu associated with a highlighted node in the project window.

4.4.3 Messages window

View | Messages makes the messages window of the current project active. It shows messages under the headings of Make, Debug, Grep and Script.

5. Setting options within a project

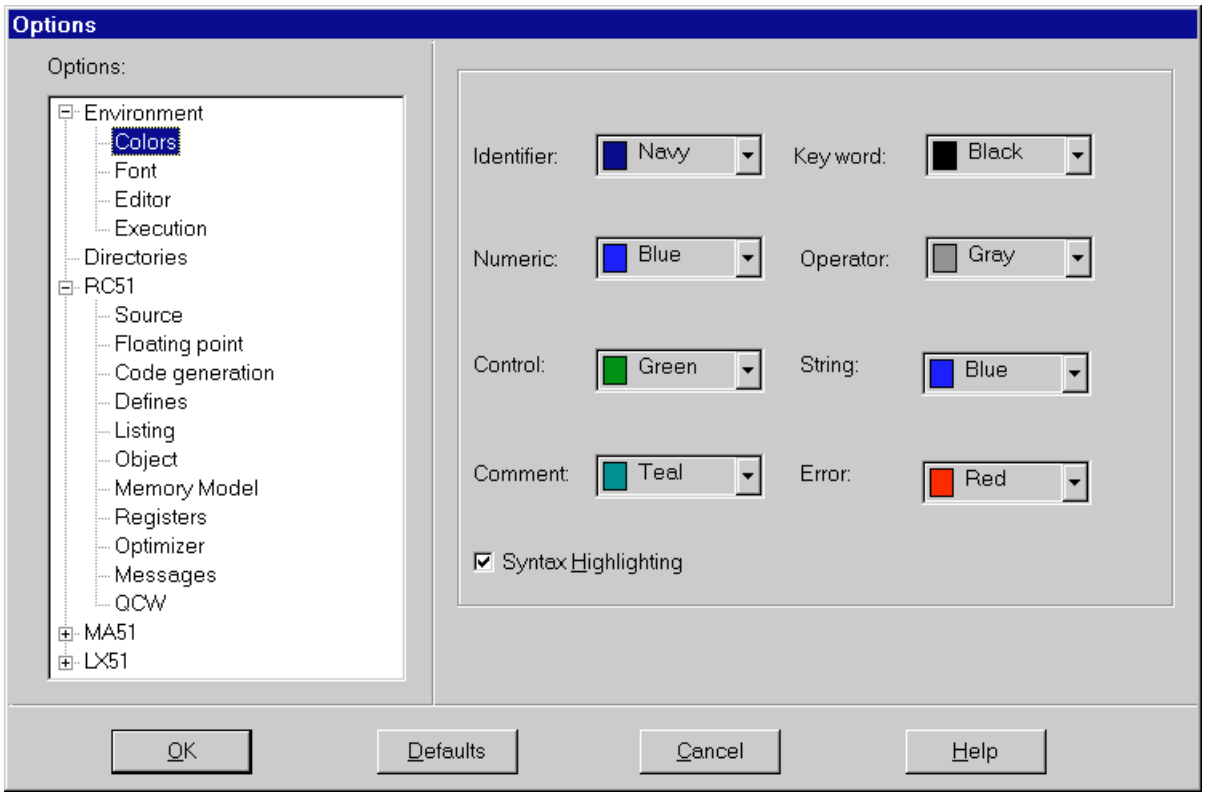
5.1 Project Options

You can tailor the behaviour of the tools to meet specific needs of your project. Options may be set either from the **Options** menu or from a pop-up menu that appears when the right mouse key is pressed with a node highlighted in the project window. Generally, the main **Options** menu should be considered as a means of defining global options with the pop-up menus used to define local variations on a node by node basis.

The options that appear via the **Options** menu depend to some extent on the type of project/processor that has been selected. For example, the 80C51 and 80C51XA have different memory models, but in some other respects, options are similar. The description below deals first with the 80C51 and then the 80C51XA.

Some options are set via 'pragmas' in the source files and these have priority over global or local options.

For project options select **Options | Project** and a window like the one below will appear.



Typical Project Options box

The left-hand part shows a list that is similar to a list of folders in Windows Explorer and so when the folder icon contains a plus symbol it may be expanded. In fact, the figure above shows the 'tree' of options when the 'Environment' and 'Compiler' folders have already been expanded for an 80C51 project. The right-hand part varies according to which line is highlighted and currently shows the options available for colour coding of sections of the source file(s).

Extensive context-sensitive on-line help is available that describes the meaning of each of the options and is easily engaged by pressing the 'F1' key when a line is highlighted.

5.2 Environment Options

There are 5 types of options associated with the general operating environment.

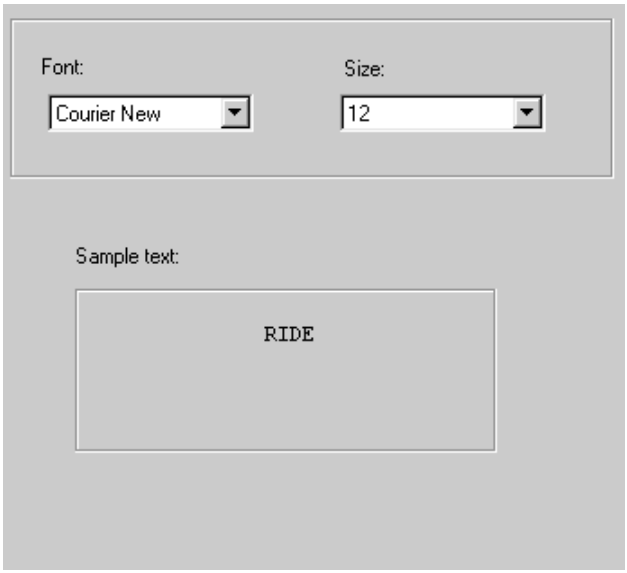
5.2.1 Colours

RIDE uses an editor that can highlight different syntactic elements, such as strings or operators, in different colours. This facility helps to identify different parts of the source code and track down any syntactic errors. However different people usually prefer to use different colours and **RIDE** allows the relationship between a syntactic element and the colour in which it is shown, to be configured by the programmer. There is a choice of 16 colours and if the syntax highlighting box is not checked colour coding will not be used. These changes are global to a project and affect all source files with extensions recognised by **RIDE**.

The dialogue box is shown above and obtained by selecting **Options | Project | Environment | Colours**.

5.2.2 Font

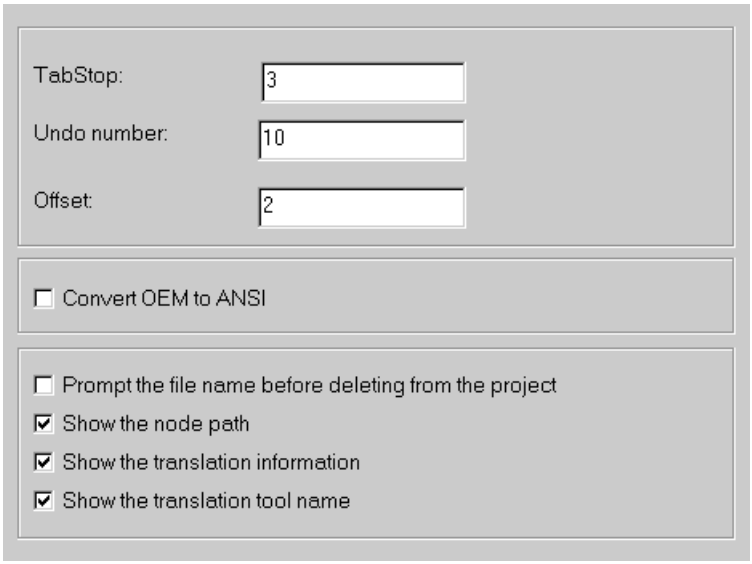
The font type and size may be selected by **Options | Project | Environment | Font** to give a window where the right-hand portion appears thus :



Font selection window

5.2.3 Editor

The behaviour of the editor may be controlled via **Options | Project | Environment | Editor,** which gives a window where the right-hand portion appears thus :

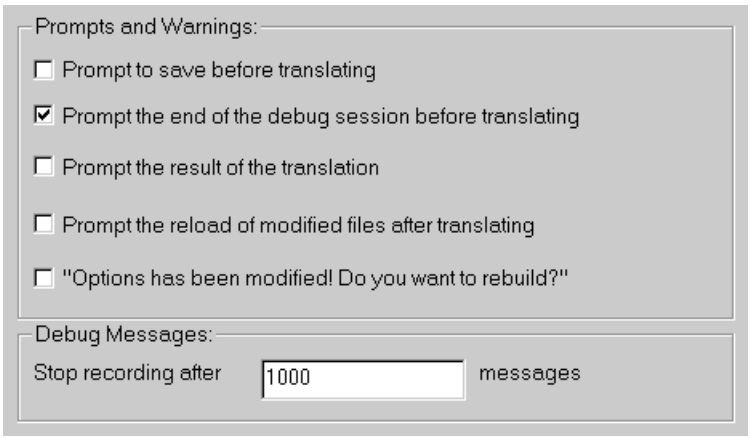


Editor Control dialogue box

- **Tab Stop** : Tab characters may be used in any source file. This entry shows the number of spaces associated with a tab character.
- **Undo number** : This input box determines the number of actions that may be undone via the **Edit | Undo** command.
- **Offset** : This input box allows the user to specify the Left Margin for edit windows in terms of the number of pixels.
- **Convert OEM to ANSI** : If this option is checked, characters are read from files as OEM characters and saved as OEM characters. When editing the file they are converted to ANSI in order to make them compatible with **RIDE**.
- **Prompt the filename before deleting from the project** : If this option is checked, **RIDE** displays a Message Box asking for confirmation before deleting nodes in the project window.
- **Show node path** : If this option is checked, the path for the node is shown in the project window.
- **Show translation information** : If this option is checked, translation information, such as the size of the associated code in bytes and other useful parameters, is shown in the project window.
- **Show the translation tool name** : If this option is checked, the name of the translation tool for each node is shown in the project window.

5.2.4 Execution

The general execution of the tools may be controlled via **Options | Project | Environment | Execution,** which gives a window where the right-hand portion appears thus :



Project Execution dialogue box

The execution options let you enable or disable **RIDE** to prompt with Message Boxes to signal the following events.

- **Prompt to save before translating** : If an editing window is opened and the text modified, **RIDE** will open a message box and ask for confirmation of the modification, before a Compile, Make all or Build all command. By default no message and auto-saving is active.
- **Prompt the end of the debug session before translating** : **RIDE** will open a message box and ask for confirmation to end the debugging session, before a Compile, Make all or Build all command. By default, the message box is not checked.
- **Prompt the result of the translation** : **RIDE** will open a message box and announce the result of the translation. By default, there is no message.
- **Prompt the reload of modified files after translating** : After the translation of a source file **RIDE** will open a message box and ask for confirmation before re-loading any modified file(s) from the disk. By default, there is no message and auto-reloading occurs.
- **Options have been modified! Do you want to rebuild?** : When this option is checked the programmer is prompted to rebuild the object file(s) whenever any option is changed. By default, this option is off.
- **Stop after xxx messages** : The debugger may be set to stop recording after xxx messages.

5.2.5 Directories

Although not strictly part of the 'Environment' folder, 'Directories' follows next and is related to the global environment. Using **Options | Project | Directories** it is possible to set the path(s) for the 'Include', 'Library', 'Object' and 'Listing' files.

- **Include** : This box specifies the folders/directories that contains the standard include files. Multiple paths are allowed and should be separated by semicolons.
- **Library** : This box specifies the folders/directories that contain the library files. Multiple paths are allowed and should be separated by semicolons.
- **Object** : This box specifies the folder/directory that contains the object (*.obj) files and the application (*.aof) file.
- **Listing** : This box specifies the folder/directory that contains the listing (*.lst and *.m51 and *.mxa) files.

5.3 RC-51 Compiler options for the 80C51

To set the compiler options for a 80C51 based project select **Options | Projects | RC51** from **RIDE**'s IDE and then either :

- **Source**: Options concerning source code format.
- **Floating point** : Options concerning floating point numbers.
- **Code generation** : Options concerning the code generation.
- **Defines**: Macro definitions.
- **Listing** : Options concerning the generated listing file.
- **Object** : Options concerning the generated Object file.
- **Memory Model** : Options concerning the memory model
- **Registers** : Options concerning the use of registers.
- **Optimizer** : Determines optimisation for speed versus code size.
- **Options Messages** : Options concerning the messages generated during the compilation.
- **QCW** : Is not useful for developer. It is an integer which defines the main options of RC51. It may be used by the support team for recreating the configuration of an existing project.

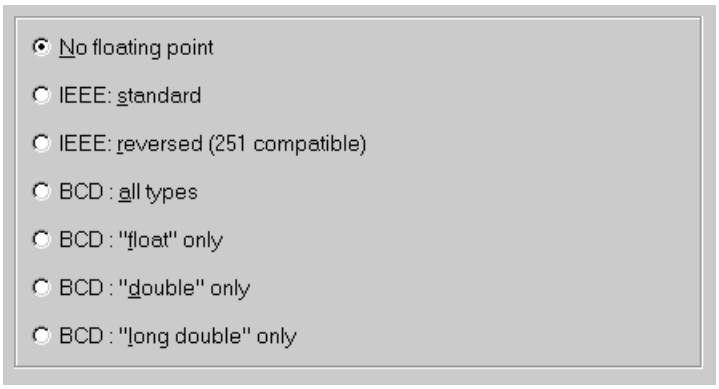
5.3.1 Source options

RC-51 Source options

The source options tell the compiler how to recognise keywords in the program :

- **ANSI C** : Allows compilation in ANSI standard. By default, this option is chosen.
- **80C51 specific language extensions** : Adapts the ANSI standard to make better use of the 80C51's architecture. In most case, this mode will be the more efficient than native ANSI but may result in lack of portability.
- **'Struct/union/enum' optional** : For a variable declaration of a struct, union or enum type, the keyword struct, union or enum can be omitted. By default, this option is not checked. (For C, 'struct' or 'union' must be used to declare a structure. For C++, these keywords may be omitted (as for classes) when the structure has been previously declared e.g. struct sPoint { int x ; int y ; }; sPoint p1 = {1,2}; However, these omission are not ANSI-C compliant, and should be sometimes confusing. The option allows to authorise or not the omission.)

5.3.2 Floating-Point options

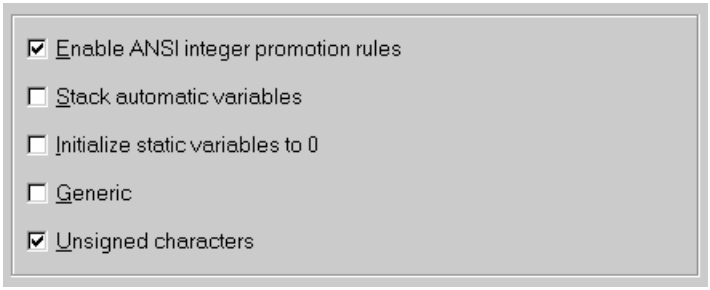


RC-51 Floating Point options

These different options specify how the compiler handles floating-point numbers. The speed of execution, code size and accuracy are all affected by the choice of format.

- **No Floating Point** : Do not use any floating point. This is the default setting.
- **IEEE** : **standard** : The standard IEEE format (" little endian ") with 23-bit unsigned mantissa, 1-bit sign and 8-bit exponent including sign.
- **IEEE** : **reversed (251 compatible)** : The standard IEEE format with 23-bit unsigned mantissa, 1-bit sign and 8-bit exponent, but stored in reverse order to the standard.
- **BCD**: all types have 6-bit unsigned exponent, 1-bit mantissa sign and 1-bit exponent sign. Mantissa size for 'float' is 24-bits, for 'double' is 40-bits and for 'long double' is 48-bits, corresponding to 6, 10 and 12 digits respectively. Note that BCD and IEEE cannot be mixed (the selection is exclusive).

5.3.3 Code generation options



RC-51 Code Generation options

- **Enable ANSI integer promotion rules** : With RC-51, the simple integer variables (int) are 16-bit words, providing compatibility with PC implementations of the C language. However, the ANSI standard requires that any character appearing in an expression shall be converted to an integer. With an 8-bit processor, this conversion (or promotion) creates additional, and therefore slower, code. The optimisation therefore consists of accepting the calculation of characters but has the disadvantage of not complying with the standard. To enable/disable the promotion, use this compiler option. If this option is checked, char to int promotions will be performed according to the standard, otherwise promotions between characters will not be performed. By default, this option is checked.
- **Stack automatic variables** : If this option is checked, the automatic variables are stacked. For the **SMALL** model, the microcontroller's internal stack is used but for the

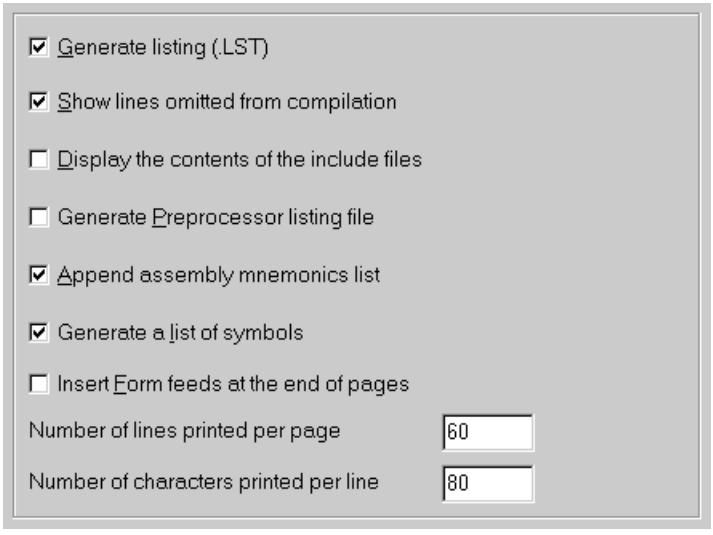
LARGE model, an external stack is used. Otherwise, the normally automatic variables are placed in a overlayable static segment (in **DATA** for the **SMALL** model and in **XDATA** for the **LARGE** model). The ‘ overlayable ‘ attribute indicates that the linker can overlay different segments of this type at similar addresses, providing that the functions corresponding to these segments do not call each other. By default, this option is checked..

- **Initialize static variables to 0** : The ANSI standard requires that a global or static variable must be initiated to 0 when no initiator is specified. By default, this option is checked to conform to the ANSI standard, but when unchecked smaller code is generated.
- **Generic** : This option forces the « Generic » attribute on unqualified pointers. If this option is selected any pointer without a space qualifier will be used as a generic pointer, otherwise any pointer without a space qualifier will be considered as pointing to the default space (XDATA for the LARGE and HUGE models, and DATA for the SMALL and TINY models). It is recommended to remain in the same mode. Although it is easier to work with the Generic mode, it is less efficient in terms of code size and speed of execution. By default, this option is checked.
- **Unsigned characters** : This option forces the CHAR type variables to be unsigned. By default, this option is checked.

5.3.4 Defines option

The defines option allows the definition of simple macros (without arguments) which could have been defined by the pre-processor using the #define command. It allows compilation to be controlled by the project’s options rather than by modifying a file. For example #ifdef allows conditional compilation. Different macros must be separated by a semicolon and assignments are or the form DEF or DEFINE(NAME=VALUE) or DEF or DEFINE(NAME/VALUE)

5.3.5 Listing options

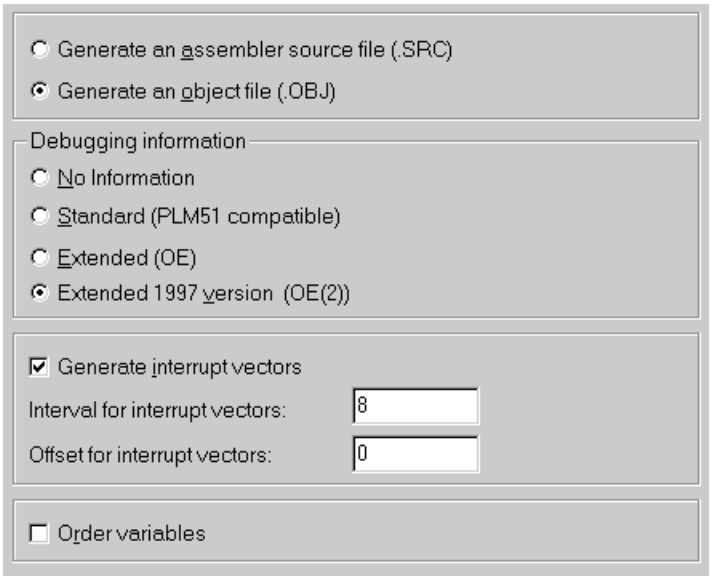


RC-51 Listing options

- **Generate Listing** : If this option is selected the compiler will produce a listing of each compiled file of the form *.lst.
- **Show lines omitted from compilation** : This option forces lines omitted from the compilation to appear in the listing file.
- **Display the contents of the include files** : This option forces the content of the include files to appear in the listing file.

- **Append assembly mnemonics list** : This option causes the assembly code generated by the compiler to be included in the listing file.
- **Generate a list of symbols** : This option causes a table of used symbols to be included in the listing file.
- **Insert Form Feeds at the end of pages** : This option allows the insertion of ASCII form-feed codes in the listing.
- **Number of lines printed per page** : Determines the lines per page.
- **Number of characters printed per line** : Determines the maximum number of characters that may appear in the listing file before the automatic inclusion of a carriage return code.

5.3.6 Object options



RC-51 Object file options

- **Generate an assembler source file** : If you choose this option, the compiler will generate an assembler source file of the form *.src, which represents the machine mnemonic equivalent of the 'C' source. The *.obj file will not be generated.
- **Generate an object file** : If you choose this option, the compiler will generate an object file of the form *.obj, which contains the 80C51 binary equivalent of the 'C' source . The *.src file will not be generated.

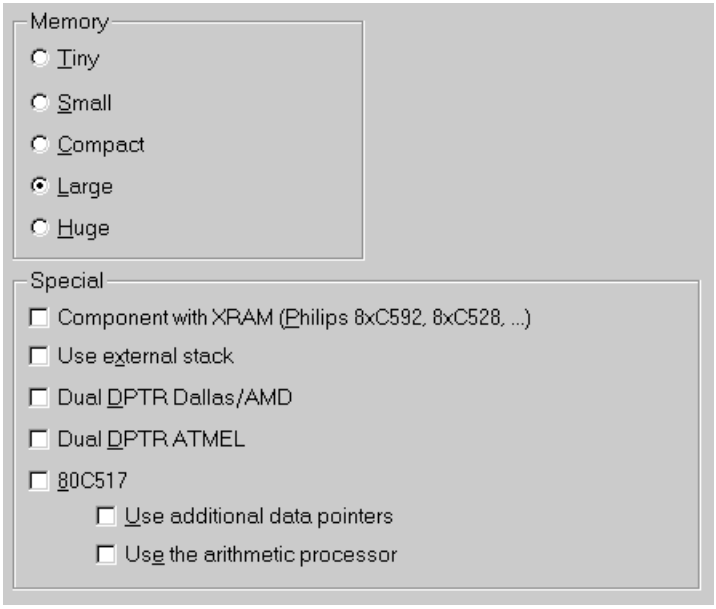
Debugging information : These options enable or disable the generation of symbol tables and line numbers in the *.obj file(s).

- **No information** : The Object file will not contain Debugging information.
- **Standard** : These tables can be retrieved by any **OMF-51** compatible development tool.
- **Extended (OE)** : If the OE option is set, the generated file will contain additional information such as the name of the source file, the types of the C variables used. High-level debugging is then possible using the SIMICE-51 simulator, PCE-51 in circuit emulator or MINI-ICE. In this case the format generated is titled « OMF-51 extended ».
- **Extended 1997 (OE2)** : New 1997 OE version, named OE2. This option allows the local variables to be watched in the debugger. It also allows the **View | Stack** command when debugging.
- **Generate Interrupt vectors** : This option instructs the compiler to generate interrupt vectors for functions that require them; An offset may be entered if the vector table starts

at an address other than 0. Using these options, the compiler generates an interrupt vector entry using either an AJMP or LJMP instruction. Vectors are located at absolute address *(interval * n)+offset+3.*

- **Order variables** : If this option is available it instructs RC-51 to order all variables in memory according to their order of definition in the C source file rather than alphabetic.

5.3.7 The compilation memory model



RC-51 Memory Model options

The compilation model defines the memory type to be used for local variables and other declarations that are given without an explicit memory type, as well as the stack location.

Memory

Model	default memory type	Xdata memory requirements	stack location	Notes
TINY	data	No	idata	suitable for the Philips 83C75x processors
SMALL	data	No	idata/pdata	the allocation of variables in external data memory is still possible
COMPACT	pdata	Yes	idata/pdata	Suitable for medium applications using an 8-bit XDATA access.
LARGE	Xdata	Yes	idata/pdata	suitable for big applications
HUGE	Xdata	Yes	pdata	suitable for the most complex of applications. Return addresses are stored in xdata memory.

Special

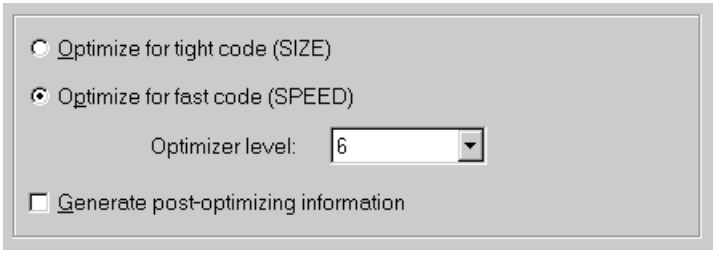
- **Component with XRAM** : Click here when using components including internal XDATA.
- **Use external stack** : The recursive features of 'C' require that local (non-static) variables are managed dynamically. As the microcontroller's internal stack is limited by the size of the internal RAM, it is necessary to use an external stack in eXternalDATA space for LARGE, COMPACT or HUGE model applications. However, the difficulty of manipulating 16-bit words has resulted in a more appropriate solution for the 80C51 architecture. Each stack can contain up to 256 bytes and is identified by an 8-bit address (via the R0 or R1 registers, using port P2 and the 'movx @Ri, A' or 'movx A,@Ri' instructions). The HUGE model implies the use of external stack. However, external stack use for SMALL, COMPACT and LARGE models is optional. The declaration of external stack use is done with the EXTSKT control.
- **Dual DPTR Dallas/AMD** : This option instructs the compiler produce code for the dual data pointers of the AMD 80C521 and compatible derivatives.
- **Dual DPTR ATMEL** : This option instructs the compiler produce code for the dual data pointers of 80C51 microcontrollers made by ATMEL.
- **80C517** : These options instruct the compiler RC-51 to produce code for the additional arithmetic processor and/or the data pointers of the Siemens 80C517.

5.3.8 Registers options

RC-51 Register options

- **Use absolute register addressing for R0-R7** : This option permits the compiler to use absolute or memory mapped addressing for registers R0 to R7.
- **Pass function arguments in registers** : This option enables the compiler to use registers for (up to three) function arguments
- **Register bank** : This option instructs the compiler which register bank to select for functions.

5.3.9 Optimisation options

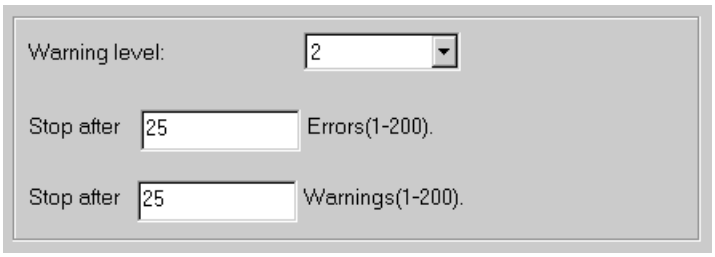


RC-51 Optimisation options

- **Optimize for tight code (SIZE)** : If this option is checked, the compiled code is optimised for Size.

- **Optimize for fast code (SPEED)** : If this option is checked, the compiled code is optimised for speed using a 1 of 7 optimiser levels. Please refer to the compiler documentation for more details.
- **Generate post-optimizing information** : Raisonance is going to launch a new Linker that performs a global, inter-modules optimisation. To allow this task, both the compiler and the linker have to generate some specific information. The option must be checked to allow the linker to optimise.

5.3.10 Compilation messages options



RC-51 Messages options

- **Warning level** : Warnings are generated at 2 levels and this option determines which levels are reported. It does not affect Error messages.
- **Stop After xxx errors** : This option makes the compiler stop after the specified number of errors has been reported.
- **Stop After xxx warnings** : This option makes the compiler stop after the specified number of warnings has been reported.

5.3.11 QCW pragma

RC-51 QCW pragma

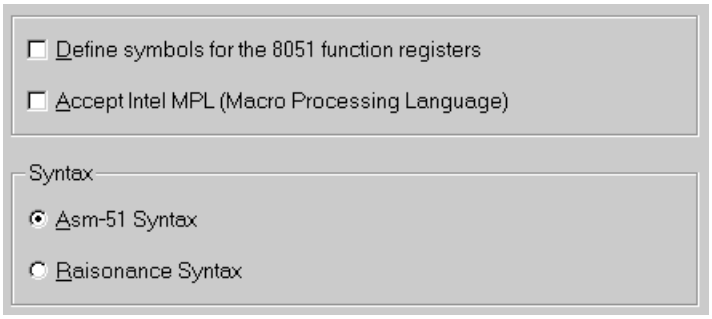
QCW is a integer that defines the main options of RC51. It may be used by the support team for simply recreating the configuration of an existing project.

5.4 MA-51 Macro Assembler options for the 80C51

To set the assembler options for a 80C51 based project select **Options | Project | MA51** from **RIDE**'s IDE and highlight either :

- **Source** : Options concerning source code format.
- **Set** : Opportunity to set the values of symbols.
- **Listing** : Options concerning the generated listing file.
- **Object** : Options concerning the generated object file

5.4.1 Source options



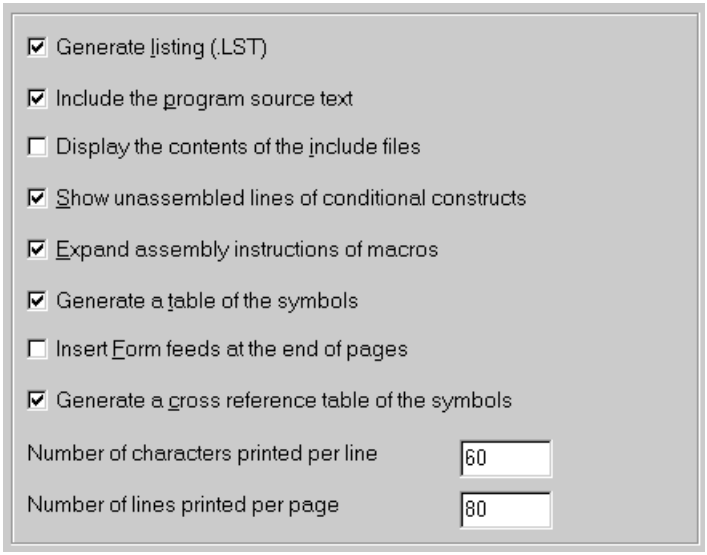
MA-51 Source options

- **Define symbols for the 8051 function registers** : This option allows the assembler to define symbols for the 80C51 Special Function Registers. By default, this option is not checked.
- **Accepts Intel MPL** : If this option is set MA-51 recognises and processes Intel ASM51 compatible macro definitions and invocations. By default, this option is not checked.
- **Asm-51 Syntax** : This is the default setting and makes the assembler accept 'standard' 80C51 assembly syntax as defined by Intel.
- **Raisonance Syntax** : This setting is mutually exclusive with respect to the **Asm-51** option and when checked the assembler accepts an extended syntax defined by Raisonance SA. This syntax allows for greater compatibility with the 80C51XA.

5.4.2 Set options

- **Set** : The Set input box permits the assignment of a numerical value or register symbol to a specified symbol name. Multiple set statements are separated with semicolons and assigned values with an equal sign (=). This option is included to allow rapid modifications to a project while avoiding changes to the associated source files.

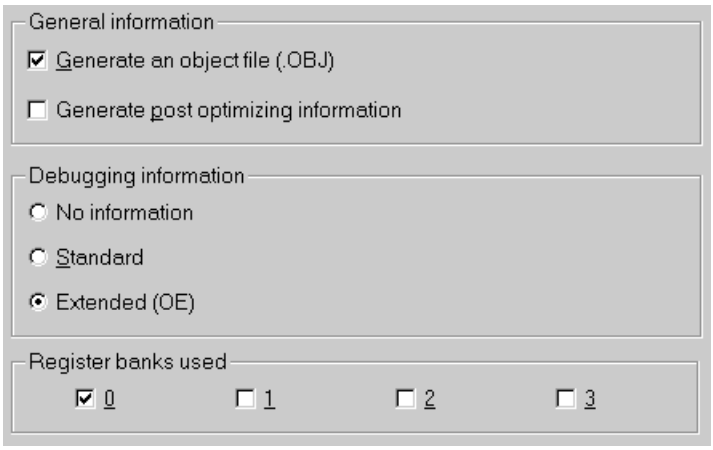
5.4.3 Listing options



MA-51 Listing options

- **Generate Listing (.LST)** : If this option is selected the assembler will produce a listing for each assembled file in the form *.lst. By default, this option is checked.
- **Include the program source text** : When selected this option instructs the assembler to include the program source text in the generated listing file. By default, this option is checked.
- **Display the contents of the include files** : This option forces the content of the include files to appear in the listing file. By default, this option is not selected.
- **Show unassembled lines of conditional constructions** : MA-51 supports conditional assembly by replacing the appropriate source-code lines with a blank line. If this option is checked, the lines of source that are not assembled in a conditional structure will be included in the listing file. By default, this option is checked.
- **Expand assembly instructions of macros** : If this option is checked the MA-51 assembler includes macro expansion text in the listing file, otherwise only the macro name is listed. By default, this option is checked.
- **Generate a table of symbols** : This option causes a table of used symbols to be included in the listing file. By default, this table is generated.
- **Generate a cross-reference table of the symbols** : This option directs the assembler to include the cross-reference table of the used symbols in the listing file. By default, this table is not included.
- **Insert Form Feeds at the end of pages** : This option allows the to insertion of ASCII form-feed codes in the listing.
- **Number of lines printed per page** : Determines the lines per page.
- **Number of characters printed per line** : Determines the number maximum number of characters that may appear in the listing file before the automatic inclusion of a carriage return code.

5.4.4 Object options



MA-51 Object options

General Information

- **Generate an object file (.OBJ)** : This option will make the assembler generate an object file. If this option is not checked, only the listing file is created. By default, this option is checked.
- **Generate post optimizing information** : Raisonance is going to launch a new Linker that performs a global, inter-modules optimisation. To allow this task, both the compiler and the linker have to generate some specific information. The option must be checked to allow the linker to optimise

Debugging information

- **No information** : No debugging information is included
- **Standard** : This format complies with the format used by the Intel PL/M-51 compiler.
- **Extended** : This option directs the assembler to include additional debugging information in the object file, which is used by Raisonance's debugging tools for testing the program.. This format complies with the format used by the Keil C51 compiler (pragma OE). It's a superset of the previous one. By default, this option is checked

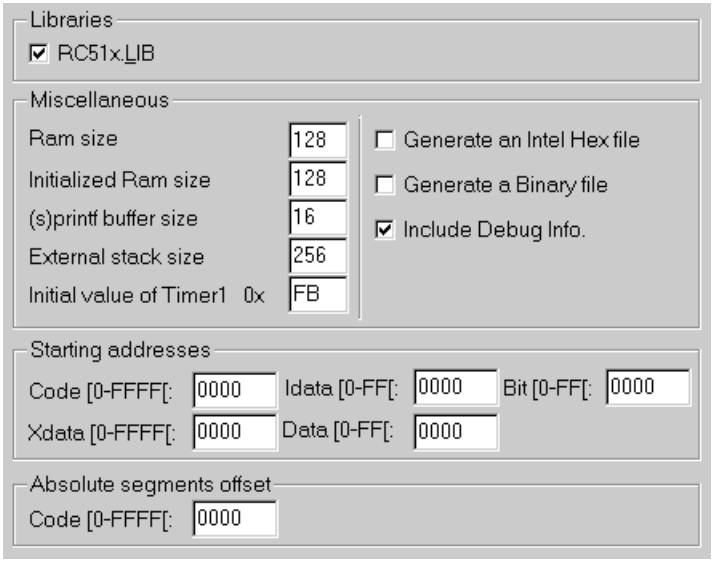
Register banks used : This list of check boxes specifies the register banks used in the source module.

5.5 LX-51 Linker options for the 80C51

To set the linker options for a 80C51 based project select **Options | Project | LX51** from **RIDE**'s IDE and highlight either :

- **Linker** : Determines various general options.
- **More** : Additional linker commands and definitions.
- **Bank switching** : Controls memory bank switching.
- **Kernel** : Allows a Real-Time kernel to be linked with certain properties.
- **Listing** : Specifies the format of the listing file.
- **Flash** : Special options for Flash memories.
- **Monitor** : Defines monitor options and especially serial communications

5.5.1 Linker options



LX-51 Linker options

Libraries

- **RC51x.LIB** : check this option to include the main 'C' compiler libraries. For projects written entirely in assemble this library is not usually needed.

Miscellaneous

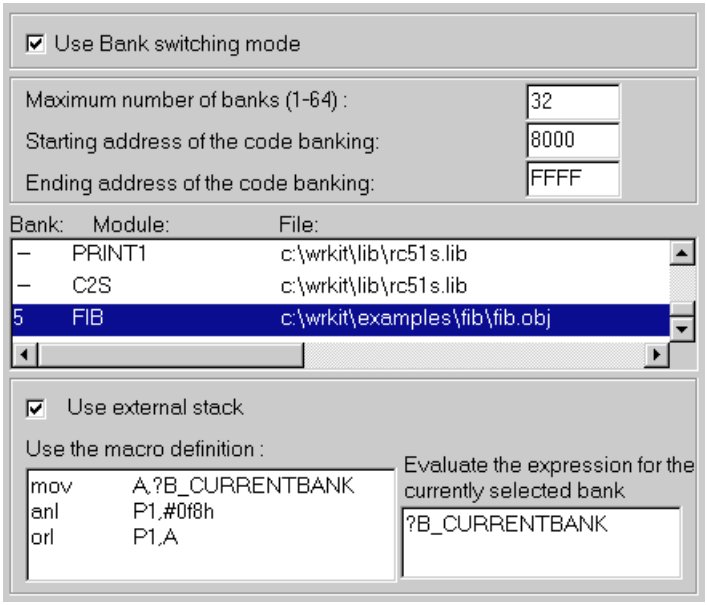
- **Ram Size** : is used to define the size of the internal DATA space. It concerns both the "data and idata" qualified variables. Internal XDATA (XRAM) is NOT affected by this number.
- **Initialized Ram size** : The startup routine (see source file in LIB\SOURCE) will set the first N bytes of the internal DATA-IDATA memory to 0. N has to be specified here.
- **(s)printf buffer size** : 'printf' is a recursive function that requires a buffer taken in the stack to format strings (if any). This buffer can be resized in case of large strings, or reduced if only short integers are formatted.
- **External stack size** : The external stack is taken from the pdata space (Port2 is set in the startup). It can be resized (from 0 to 256).
- **Initial value of Timer 1** : When either 'printf' or 'putchar' is used in the program, a special startup is selected that initialises the UART. It's why <<printf ("Hello World!");>> works without anything else. By default, Timer1 is used as baud rate generator (in 8-bit auto-reload mode), and the reload value (0E6H by default) can be modified to change the baud rate. Please refer to your preferred 8051 data book to calculate the value you need.
- **Generate an Intel Hex file** : Generates an ASCII file of the form *.hex in Intel's Hex file format. This is useful for downloading to some device programmers.
- **Generate a Binary file** : Generates a binary file of the form *.bin.
- **Include Debug Info** : places additional information in the *.aof file to aid subsequent debugging. Also a file without an extension is generated to conform with Intel PL/M-51 output naming conventions)

Starting addresses : Allows the starting addresses for different address spaces to be defined.
Absolute Segments Offset : If, for example, your hardware has 32KB of RAM starting at address 8000H and nothing from 0 to 8000H, you have to indicate an offset of 8000H to relocate the XDATA segments after 8000H .

5.5.2 More controls

This allows linker commands to be added in a manner similar to pragmas within a source file, RC-51 Defines, or MA-51 Set options. For example, 'XDATA (8000H)' will relocate the relocatable XDATA segments after 8000H. In this case, the directive can be replaced by indicating 8000H in the field "code offset".

5.5.3 Bank Switching



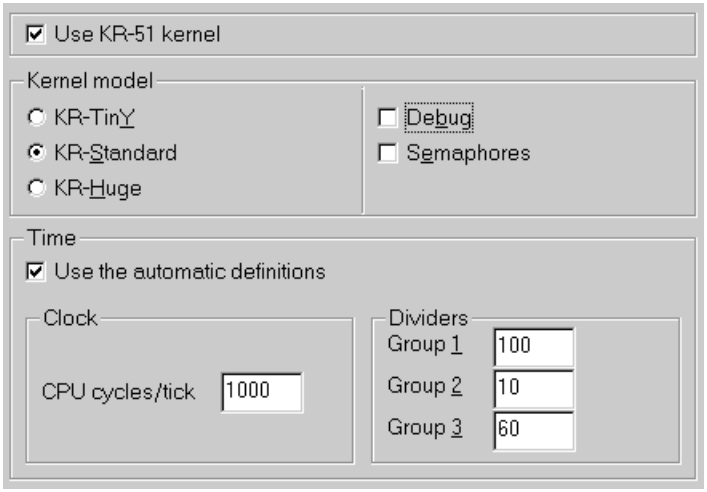
RX-51 Bank Switching options

Bank switching is used in some applications where the inherent address range of the particular 80C51 microcontroller is insufficient. Typically, Port P1 is used to provide additional address/control lines that select a particular 'bank' of memory. Naturally, the options must be given values that correspond exactly with the actual hardware. The default setting is for the Bank switching option is unchecked and therefore all other options are unavailable. When checked the following applies :

- **Maximum number of banks** : set the maximum number of banks from 1to 64.
- **Starting address of the code banking** : This number determines the start address of the page. Since the 32K-32K mode is taken by default, the starting address is set to 8000h.
- **Ending address of the code banking** : This number determines the end address of the page. Since the 32K-32K mode is taken by default, the ending address is set to 0FFFFh.
- **The list of the project's modules** : This list contains the different modules of the project, and lets the user specify the pages where the different modules will be placed. In order to select a page double-click the module name and a small edit window appears that lets you specify the bank number.
- **Use external stack** : There is an overhead associated with bank switching and selecting this option minimises the use of internal RAM but increases execution time. For more details see the description provided via on-line Help.
- **Use the macro definition** : In this edit window, the user must specify the decoding macro adapted to the associated hardware design.
- **Evaluate the expression**. : This expression must correspond with the preceding macro and it is used by the debugger to calculate the current bank number.

The project manager automatically adds ..\Ride\Lib\L51_bank.a51 to the project to manage bank switching. See the LX-51 Reference Manual for additional information and in case of difficulty contact **Raisonance SA.**

5.5.4 Kernel



RX-51 kernel options

If the **Use KR-51 kernal** box is checked, the following options are available :

Kernel Model

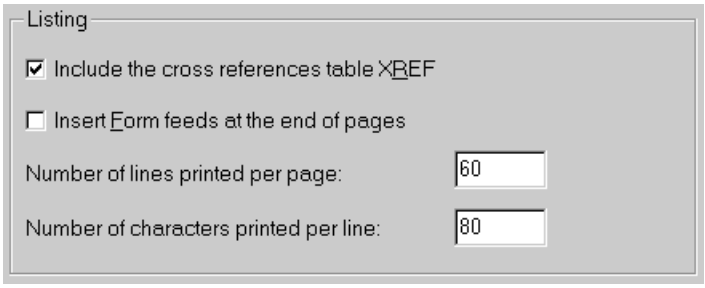
- **KR-TinY** : up to 8 tasks, use of DATA only.
- **KR-Standard** : up to 32 tasks. XDATA space is required.
- **KR-Huge** : up to 256 tasks. XDATA space is required.
- **Debug** : use the extended "debug" libraries of the Kernel to help in debugging a crash.
- **Semaphores** : use of the larger "semaphore" libraries is mandatory if services using semaphores are called.

Time

- **Use the automatic definitions** : This option must be checked for the subsequent definitions to be used, otherwise the kernel timing must be specified by an assembly file. This option is checked by default.
- **CPU cycles/tick** : Determines the kernal's tick rate with respect to the processor's clock.
- **Dividers for Group 1, 2 and 3** : These determine secondary clocks. The three dividers define task groups that permit faster task examination and better general efficiency.

5.5.5 Listing

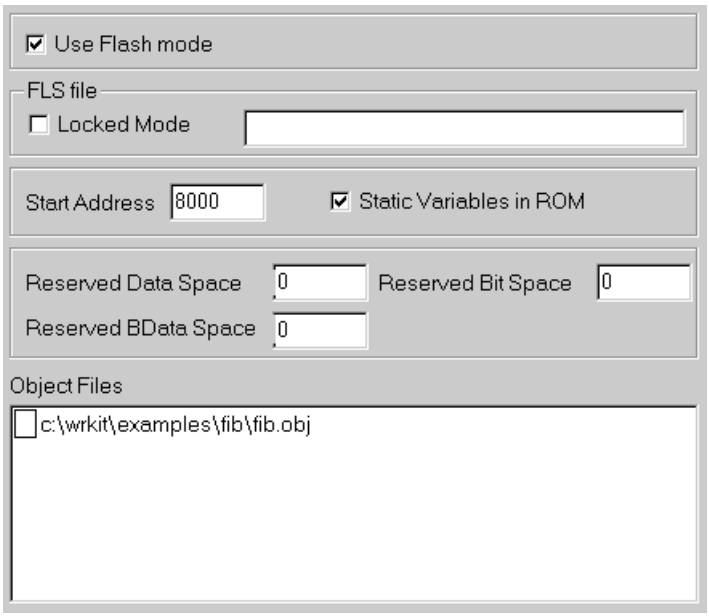
The listing options specify the format of the linker's listing file pages.



LX-51 Listing options

- **Include the cross reference table XREF** : Causes the linker to generates a cross-reference file of the form *.xrf. This option is checked by default.
- **Insert Form feeds at the end of pages** : This option allows the to insertion of ASCII form-feed codes in the listing.
- **Number of lines printed per page** : Determines the lines per page.
- **Number of characters printed per line** : Determines the maximum number of characters that may appear in the listing file before the automatic inclusion of a carriage return code..

5.5.6 Flash



LX-51 Flash options

If the **Use Flash mode** box is checked, the following options are available :

FLS file

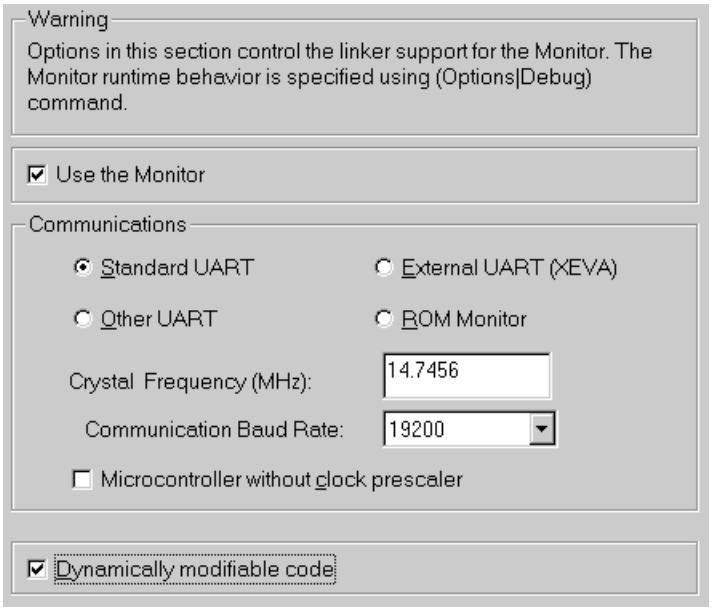
- **Locked Mode** : When "Checked", the "ROMed" part is considered has being locked and not subject to future change. Only the "Flashed" part will be modified.
- **Static Variables in ROM** : When "Checked", the code segment that is used to initialise the global/static variables (during start-up) will be located into the ROM instead of the

Flash. It must be checked when the application should be able to start without any FLASH content.

Reserved Memory Spaces : Allows reservation (for future needs) of some data (often bit addressable) before "locking" the ROM part. This 'reserve' can be used to locate future bit segments for example.

Object Files to be in FLASHed ? : When checked, a file will be located in the "FLASH" part.

5.5.7 Monitor options



LX-51 Monitor Options

If the **Use the Monitor** box is checked the following options, which affect communications, become available :

Communications

- **Standard UART** : UART0 of the 80C51 (used with Timer1 as baud rate generator)
- **Other UART** : If this option is available, it relates to UART1 of the DS320 system.
- **External UART (XEVA)** : Makes the ROM monitor use the external UART on the XEVA-DEMO (Raisonance's Universal evaluation board).
- **ROM Monitor (customized)** : Means that the I/O part of the ROM monitor has been defined by the user.
- **Crystal Frequency** : The frequency of the crystal attached to the microcontroller
- **Communications Baud Rate** : The required Baud rate
- **Microcontroller without clock prescaler (x2 like 8xC51RB2)** : It means that the 8051 core runs with a 6 clock cycle instead of 12.

Dynamically modifiable code : Means that the CODE will be contained in RAM, that allows breakpoints to be set and the code edited. Note this feature is mandatory for support of breakpoints.

5.6 RC-XA Compiler options for the 80C51XA

The options for the 80C51XA compiler are similar to those for the 80C51, which have already been described in section 5.3. However the additional addressing range and capabilities of the 'XA necessitates extended and/or additional options in certain areas.

To set the compiler options for a 80C51XA based project select **Options | Projects | RCXA** from **RIDE**'s IDE and then either :

- **Source**: Options concerning source code format.
- **Floating point**: Options concerning floating point numbers.
- **Code generation**: Options concerning the code generation.
- **Defines**: Macro definitions.
- **Listing**: Options concerning the generated listing file.
- **Object**: Options concerning the generated object file.
- **Memory Model**: Options concerning the memory model
- **Registers**: Options concerning the use of registers.
- **Optimizer** : Determines optimisation for speed versus code size.
- **Options Messages**: Options concerning the messages generated during the compilation.
- **QCW** : Does not concern the developer, it is an integer that defines the main options of RC51. It may be used by the support team for simply recreating the configuration of an existing project.

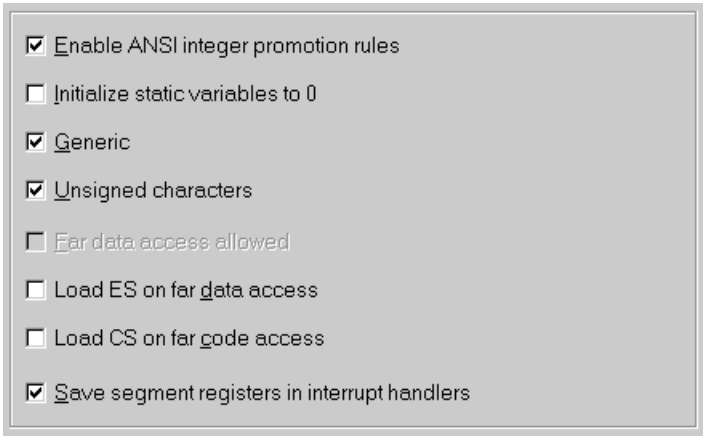
5.6.1 Source options

Same as RC-51 but 80C51 changed to 80C51XA

5.6.2 Floating point options

Similar to RC-51 but the only choice is No floating point or the IEEE standard.

5.6.3 Code generation options



RC-XA Code Generation options

Some of the following options have been introduced to optimise the code for specific architectures (like the SmartXA). They are not necessary useful for a standard XA.

- **Enable ANSI integer promotion rules** : Same as RC-51

- **Initialize static variables to 0** : Same as RC-51
- **Generic** : Same as RC-51
- **Unsigned characters** : Same as RC-51
- **Far data access allowed** : This option is available when XA Page0 mode has been chosen. If this option is selected the Data Segment Register (DS) is used to access near data and the Extra Segment Register (ES) is used to access far data. Otherwise near and far keywords are not allowed in Page0 mode. By default, this option is not checked.
- **Load ES on far data access** : If this option is not selected the Extra Segment Register (ES) is expected to contain the required fixed value. The access of far data will not update the value of this register. However if this option is checked, the ES register is updated before being used. This option is not available when XA Page0 mode has been chosen. This option is mostly used for the SmartXA when an Operating System has preset ES)
- **Load CS on far code access** : If this option is not selected the Code Segment Register (CS) is expected to contain the required fixed value. The access of far code data will not update the value of this register. However if this option is checked, the CS register is updated before being used. This option is not available when XA Page0 mode has been chosen.
- **Save segment registers in interrupt handlers** : If this option is selected, Interrupt handler routines (Exception Interrupts, Event Interrupts, Software Interrupts etc.) will save, and then restore, the Code Segment Register (CS) and Extra Segment Register (ES) values. The compiler will only write onto DS during the start-up and never during code generation. (To the XA, a different DS means a different program/task). By default, this option is checked.

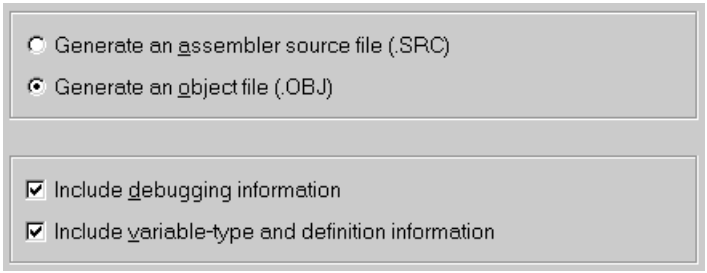
5.6.4 Defines option

Same as RC-51

5.6.5 Listing options

Same as RC-51

5.6.6 Object options

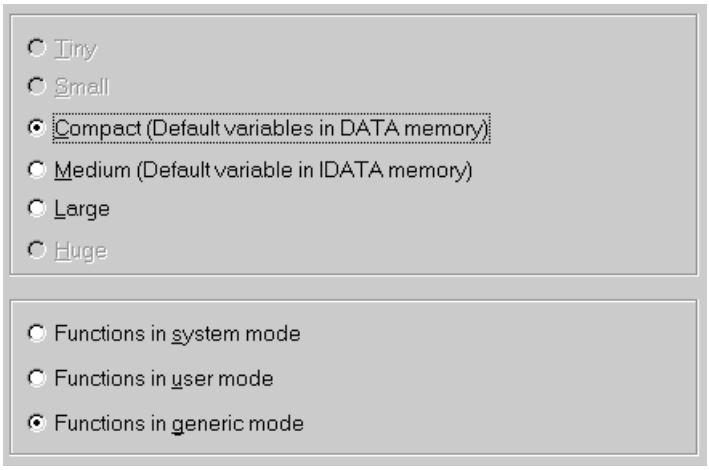


RC-XA Object file options

- **Generate an assembler source file** : Same as RC-51..
- **Generate an object file** : Same as RC-51

- **Include debugging information** : The Object file will contain Debugging information if this is checked.
- **Include variable-type and definition information** : This information is needed to make full use of the facilities within the integrated debugger and so the option would normally be checked.

5.6.7 The compilation memory model



RC-XA Memory Model options

Memory models are similar to those in RC-51 but the **Tiny** and **Small** models are available only if XA Page0 mode has been chosen. **Medium** is very similar to **Compact**, and the difference will only affect the default addressing mode for global variables: direct data or indirect idata.

Functions may be placed in either **system**, **user** or **generic** mode, which is useful for protecting different parts of code particularly in multi-tasking applications.

5.6.8 Registers options

Unlike the RC-51 there are no register options associated with the compiler.

5.6.9 Optimization options

Same as RC-51

5.6.10 Compilation messages options

Same as RC-51

5.6.11 QCW pragma

Same as RC-51 (QCW is a integer which defines the main options of RC51. It may be used by the support team for simply recreating the configuration of an existing project.

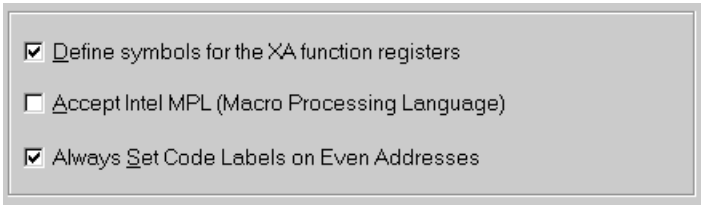
5.7 MA-XA Assembler options for the 80C51XA

The options for the 80C51XA assembler are similar to those for the 80C51, which have already been described in section 5.4. However the additional addressing range and capabilities of the 'XA necessitates extended and/or additional options in certain areas.

To set the assembler options for a 80C51XA based project select **Options | Project | MAXA** from **RIDE**'s IDE and highlight either :

- **Source** : Options concerning source code format.
- **Set** : Opportunity to set the values of symbols.
- **Listing** : Options concerning the generated listing file.
- **Object** : Options concerning the generated Object file

5.7.1 Source options



MA-XA Source options

- **Define symbols for the XA function registers** : Same as MA-51.
- **Accepts Intel MPL** : Same as MA-51.
- **Always set code labels on Even Addresses**: In the 80C51XA all jumps must be to even addresses, however data tables in code space can begin at odd addresses. When this option is checked all labels in code space are also placed at even addresses.

5.7.2 Set options

Same as MA-51.

5.7.3 Listing options

Same as MA-51.

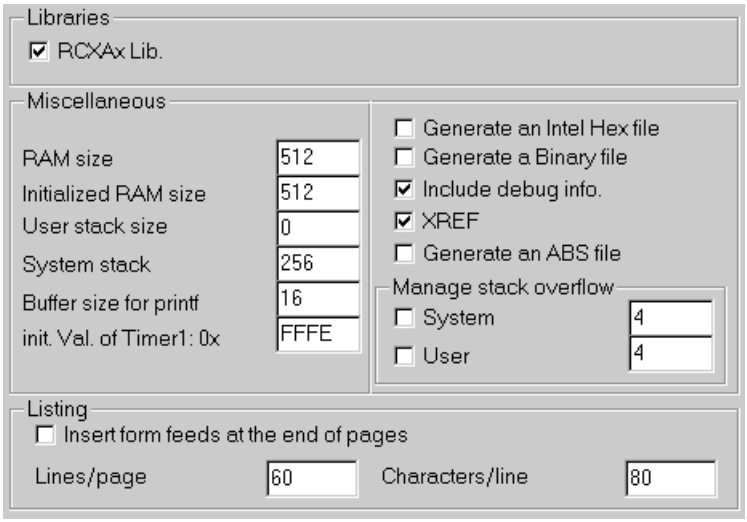
5.8 RL-XA Linker options for the 80C51XA

The options for the 80C51XA linker are similar to those for the 80C51, which have already been described in section 5.5. However the additional addressing range and capabilities of the 'XA necessitates extended and/or additional options in certain areas.

To set the linker options for a 80C51XA based project select **Options | Project | RLXA** from **RIDE**'s IDE and highlight either :

- **Linker** : Determines various general options.
- **More** : Additional linker commands and definitions.
- **Kernel** : Allows a Real-Time kernel to be linked with certain properties.
- **Monitor** : Defines monitor options and especially serial communications
- **Relocation** : Defines absolute memory addresses for relocatable code

5.8.1 Linker options



LX-XA Linker options

Some options are similar to LX-51 but relocation information is moved to a separate dialogue box called Relocation. (See below)

Libraries

- **RCXAx.LIB** : check this option to include the main 'C' compiler libraries. For projects written entirely in assemble this library is not usually needed.

Miscellaneous

- **Ram Size** : is used to define the size of the internal DATA space If the program requires more space, an error will be reported. By default this value is set to 512 Bytes
- **Initialized Ram size** : Determines the amount of DATA memory space has to be initialised to 0.
- **User stack size** : Specifies the size of the user stack. This value has to be non zero if the application requires 'user mode' functions. The default value is set to 0.
- **System stack (size)** : Specifies the size of the user stack. This value has to be non-zero if the application requires 'system mode' functions. The default value is set to 256 Bytes.
- **Buffer size for printf** : The size of the buffer used for printf formatting is fixed by this value. Set it to the maximum size of any printed string. By default, this value is set to 16.

- **Initial value of Timer 1** : The initial value of Timer 1 initial value can be set with this option. When either 'printf' or 'putchar' is used in the program, a special start-up is selected that initialises the UART. It's why <<printf ("Hello World!");>> works without anything else. By default, Timer1 is used as baud rate generator (in 16-bit auto-reload mode), and the reload value can be modified to change the baud rate. Please refer to your preferred XA data book to calculate the value you need.
- **Generate an Intel Hex file** : Generates an ASCII file of the form *.hex in Intel's Hex file format. This is useful for downloading to some device programmers.
- **Generate a Binary file** : Generates a binary file of the form *.bin.
- **Include Debug Info** : places additional information in the *.aof file to aid subsequent debugging
- **XREF** : If this option is checked a table of cross-references is added to the listing file.
- **Generate an ABS file** : The ABS format is compatible with emulators made by Ceibo. (Raisonance AOF is a proprietary format, but it is supported by several emulator vendors).
- **Manage Stack Overflow** : These options specify the number of bytes to be reserve in order to manage user/system a stack overflow.

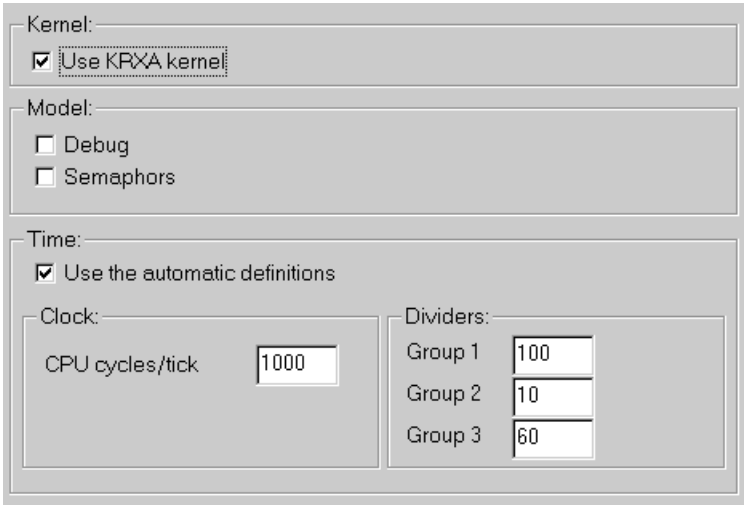
Listing

- **Insert form feeds at the end of pages** : This option allows the to insertion of ASCII form-feed codes in the listing.
- **Lines/page** : Determines the lines per page.
- **Characters/line** : Determines the maximum number of characters that may appear in the listing file before the automatic inclusion of a carriage return code..

5.8.2 More controls

This option is that same as for LX-51 and allows linker commands to be added in a manner similar to pragmas within a source file, RC-51 Defines, or MA-51 Set options. For example, 'XDATA (8000H)' will relocate the relocatable XDATA segments after 8000H.

5.8.3 Kernel



KR-XA kernel options

If the **Use KRXA kernel** box is checked kernel libraries are automatically linked to the project and the following options are available :

Kernel Model

- **Debug** : Debug libraries are to be included. The debug version includes tests for data integrity, crash traps..etc.
- **Semaphores** : Include semaphore support. The kernel without semaphore support is smaller and faster.

Time

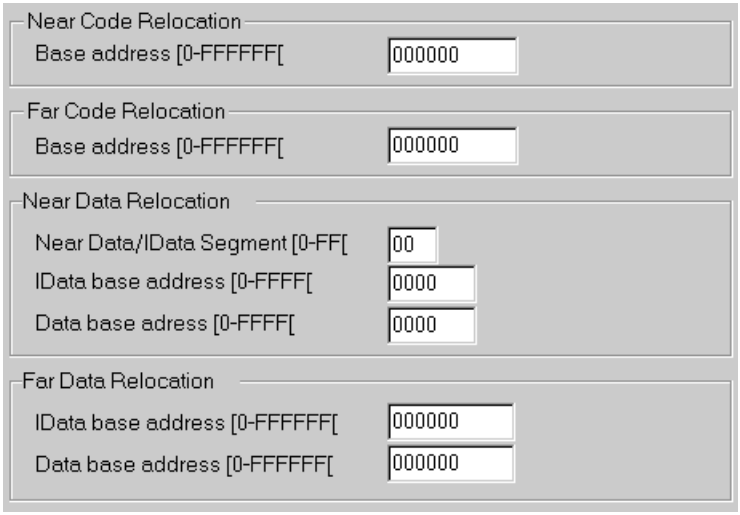
Same as KR-51.

For more details, please refer to the KR-XA manual.

5.8.4 Monitor options

Same as LX-51

5.8.5 Relocation

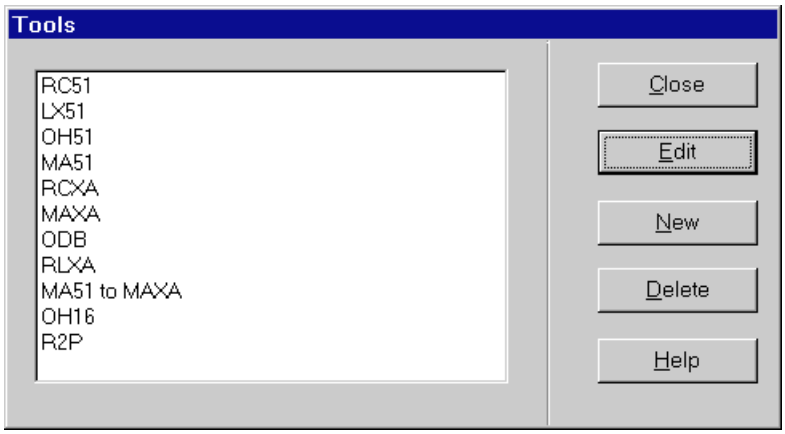


RL-XA Relocation options

The base addresses for different types of code and data are specified in a self-explanatory manner via this dialogue box. For more details, about relocation controls see the RL-XA manual.

5.9 Tools options

The software tools which are called by the IDE interface may be configured via the **Options** | **Tools** command, which opens the **Tools** dialog box.



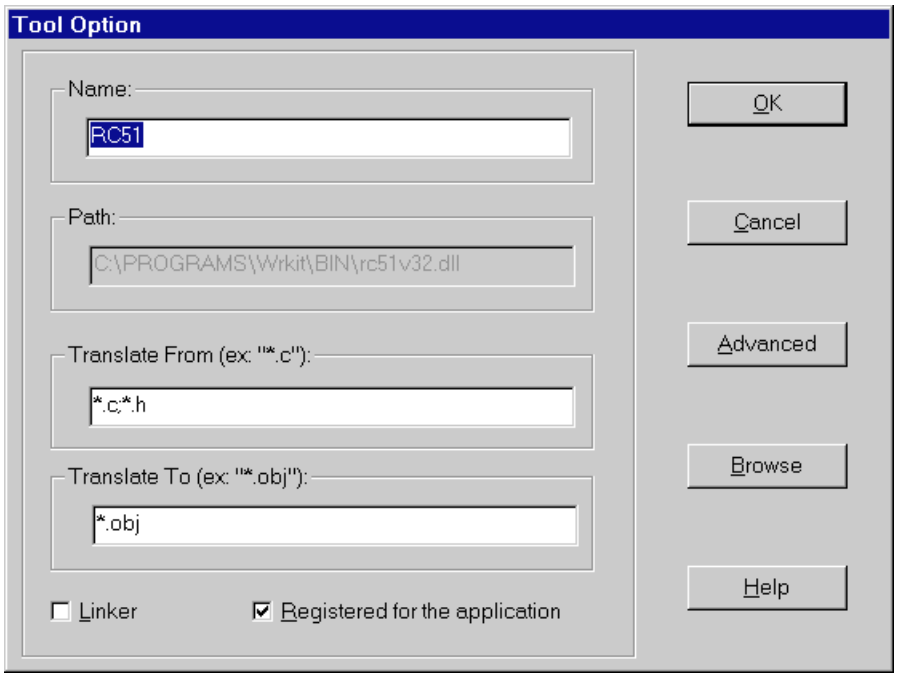
Tools selection dialogue box

The list of tools depends on exactly which tool set has been purchased and if any additional tools have been installed. For example, **RC51**, **LX51**, **OH51**, **MA51** are the pre-defined tools for an 80C51 project. A user-defined tool can be a debugging tool, a utility or a translator.

From this dialogue box there are three major options :

- **Edit** : allows the tool to be modified.
- **New** : allow a new tool to be created.
- **Delete** : highlight the name of the tool to be deleted and click this button.

Options | Tools | RC51 | Edit :

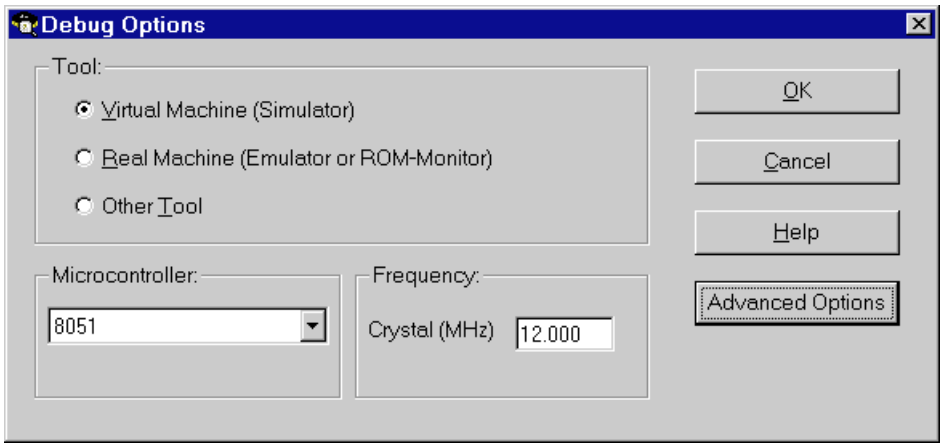


Tool Options dialogue box

- **Name** : The Name input box lets you define a name that will designate the tool in the project environment. This name will appear in tools lists. Remember that a project for the 80C51 will consider the tool named RC-51 as the default 'C' compiler, the tool named MA-51 as the default assembler and the tool named LX-51 as its linker. In the same way, a project for 80C51XA will consider the tool named RC-XA as the default 'C' compiler, the tool named MA-XA as the default assembler and the tool named RL-XA as its linker.
- **Path** : Type the complete path of the program. You can also click the **Browse** button to search for the path and filename of your program.
- **Translate From** : This input box lets you define the types of nodes that can use the tool. You can enter more than one file extension by separating them with semicolons. **RIDE** will use these extensions to select automatically the translating tool for each file. These extensions will also be used to select the appropriate syntax highlighting for the file.
- **Translate To** : Use the Translate To input box to enter the file extension for the type of file that results from applying the tool to the file types defined in the Translate From input box.
- **Linker** : Select this option if you want to use the tool as a linker.
- **Registered for the application** : With this option, you can add the defined tool to the list of **RIDE** default tools. Which means that for each new project, the tool will be automatically added to the project's list of tools. If this option is not selected the tool will remain specific to the active project. Selecting this option has no effect for existing project files. **RIDE** does not modify the old projects but will look for modifications when creating new projects.
- **Advanced Options** : This button opens an Options dialogue box similar to that described in section 5.1. However, unlike that description in which only tools applicable to the selected project are shown, this dialogue box is restricted to a selected tool, which may not be related to the current project.

5.10 Debugging options

Debugging options are configured by selecting **Options | Debug** from the main screen.



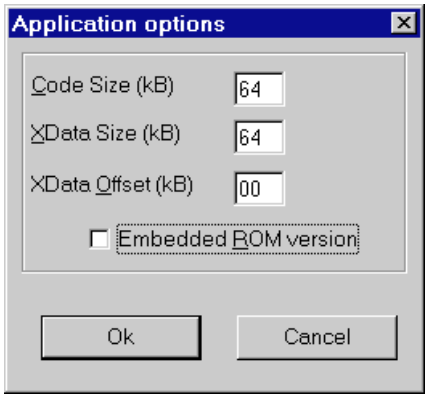
Debugging options

Tool

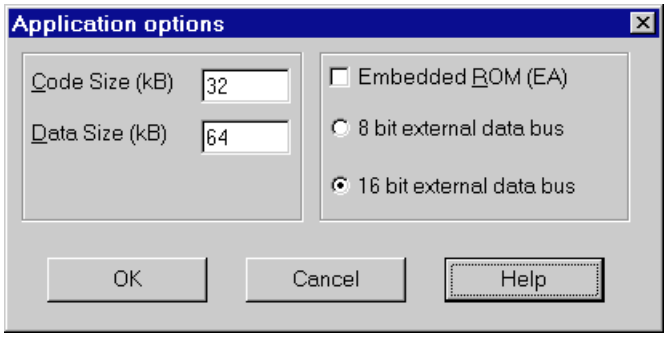
- **Virtual Machine (Simulator)** : Use the integrated simulator
- **Real Machine (Emulator of ROM-Monitor)** : Use a real emulator such as the PCE-51.
- **Other Tool** : To use another debugging tool such as **OPTOROM** EPROM Emulator. To create a new tool select **Options | Tools | New**.

One device from a list of either 80C51 or 80C51XA **Microcontrollers** must be selected, together with a specified crystal **Frequency**.

Advanced Options : The dialogue box depends on the type of debugging tool that has been selected.



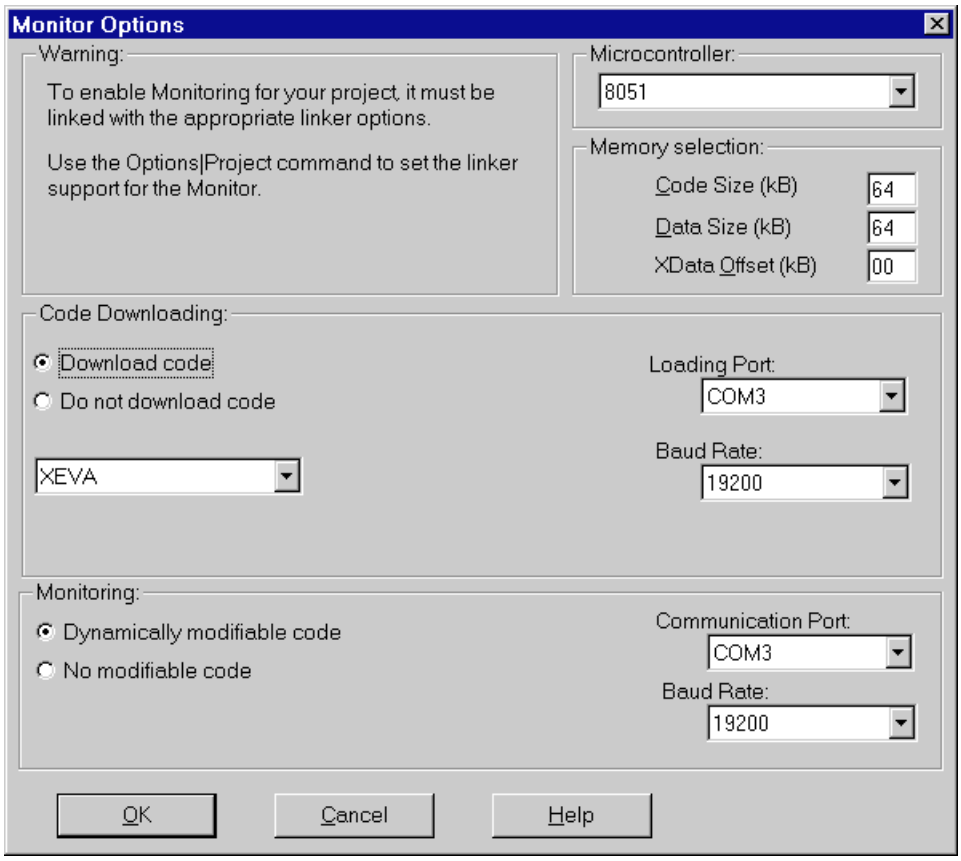
80C51



80C51XA

Advanced Options for Virtual Machines

For the Virtual Machine simulator, it is only necessary to define the size and/or offset of various data spaces and indicate if there is an embedded ROM version (although not actually used by the simulator). In addition for the 80C51XA, the width of the external data bus has to be specified.



Advanced options for Real Machine

If the user intends to debug target hardware then prior to defining the options in this dialogue box it is essential that the appropriate code has been linked into the object file(s). See **Options | Project | RX51| Monitor** or **Options | Project | RLXA | Monitor.**

Microcontroller : The most appropriate microcontroller is selected here.
Memory Selection : The code size, data size and data offset must be defined here.
Code Downloading : When **Download code** is selected, the (serial) loading port and Baud Rate may be entered. There is also a choice of type of target, currently either XEVA, OPTOROM or OTHER. If 'other' is selected then a program name and argument list has to be entered.
Monitoring : The communication port for monitoring the execution of the program must also be defined even if it the same as that used for downloading. It is common during some stages of debugging to permit bytes in the code space to be modified, for example to insert breakpoints. However this may be undesirable in some systems and the debugging environment may be configured to permit no dynamic modification of the code space.

See the chapter entitled 'Debugging' in the **'RIDE'** reference manual for more information about the advanced options and the debugging process.

6. Files, Editing and Windows Management

6.1 File Menu

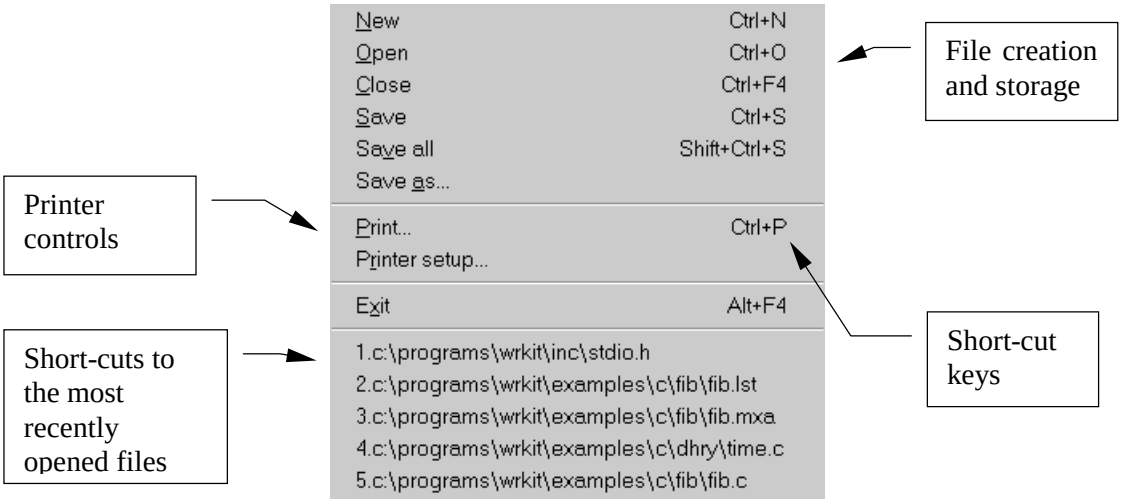
6.2 Editing Text

6.3 Window management

6.4 Command summary

6.1 File Menu

The **File** menu contains options for managing files and terminating **RIDE** and appears thus :



File Menu

6.1.1 File filters

In order to simplify the management of files of different types **RIDE** provides file filters that are used in the **Open File** and **Save As** dialog boxes. This means that only files of a particular type with user-defined extensions are visible when selecting a file from a list of files. By default the file filters are :

- **'C' Files** : filename extensions for 'C' and 'C' header files are *.c and *.h respectively.
- **Assembler Files** : filename extensions for assembler and assembler header files are *.a51 for the 80C51, *.axa for the 80C51XA, and *.inc for include files.
- **Listing Files** : filename extensions for the listing files are *.lst for compilers/assemblers and *.m51 or *.mxa for the linker.
- **All Files**: filename extensions of the form *.* represent any file.

The filename extension will determine the kind of syntax highlighting to be used when editing the file and the programming tool to associate automatically with the file.

6.1.2 Customising the file filters

To customise the filename extension associated with files of a particular type it needs to be entered into a dialogue box. For example to associate files of the form *.cx with 'C-like' source and the 80C51XA compiler select **Options | Tools | RCXA | Edit** and append **';***.cx' to the string in the section headed **Translate From**. Note that multiple file types must be separated by a semicolon. Finally choose **OK**.

6.1.3 Creating a new source file

To create a new source file :

1. Select **File | New** or press Ctrl+N.
2. From the small **Document Type** pop-up menu that appears, select the type of file. **RIDE** displays an **untitled** window with a default filename extension. As text is entered, it will be colour coded according to the file type. Note that the maximum number of characters in any line is 255.

3. To save the file select **File | Save as**, and enter the required filename with the desired extension. The directory/folder to be used should also be selected.
4. Finally, select the **Save** button.

Before an individual window can saved or closed it must be active. To make a window active, either switch to the window by clicking anywhere in it, or choose the window name or number from the **Window** menu.

6.1.4 Opening an existing source file

There are a number of ways to open an existing source file and no matter which method is used, its name is added to the list of files in the Window menu. Only one copy of a source file may be open at any time.

Short-cut :

Press Ctrl+O and follow as for step 2 below.

File menu :

1. Select **File | Open**.
2. The **Open** dialogue box appears and allows selection of the drive and directory/folder.
3. Select the types of files to display in the **Files of Type** box. Files with the chosen extension are displayed in the **File Name** box.
4. In the **File Name** box enter a filename or highlight a file from the filtered list, then press the **Open** button. Alternatively, merely double-click on the highlighted filename.

Project Window :

An existing file may also be opened by clicking on the corresponding node in the project window.

Edit Window :

Sometimes the text within a file references another file, as is the case for 'include' files. To open the referenced file, simply highlight the name and press 'Ctrl+Shift+O'. Alternatively, select **Open the Selected File** from the pop-up menu that appears by pressing the right mouse button.

Via Explorer :

Dragging the file's icon from Windows Explorer and dropping it into **RIDE** will open the file.

6.1.5 Saving a file

To save a file :

1. Make the file's edit window active by clicking in it anywhere, or by choosing the window name or number from the **Window** menu.
2. Select **File | Save** or press Ctrl+S.
3. If it is a new file in an untitled window, **RIDE** displays the **Save As** dialog box otherwise it saves to the existing filename but first preserves the contents of the original file in *.bak.
4. For the **Save as** option, select the required drive and directory/folder, enter the filename in the **File name** box and finally select the **Save** button.

6.1.6 Saving all files

In the course of developing a project, many files are created and often several are open simultaneously. To save all the open files merely select **File | Save all** or press Shift-Ctrl+S.

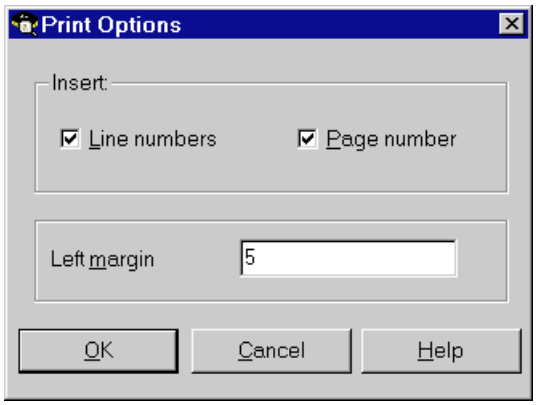
6.1.7 Closing a File

When changes to an open file have been completed the file should be closed by selecting **File | Close** or Ctrl-F4 or the close window icon ×. If changes have been made since the file was last saved, a small dialogue box appears and the user is asked to confirm that the file should be updated before the edit window is closed. It is also possible to close the window without saving changes to the file and to abandon or cancel the close command and leave the window open and active.

6.1.8 Printing files

If there is a local printer driver installed on the computer or there is a path to a network printer, the entire content of the active editing window can be printed.

1. First follow **File | Printer Setup | Printer** and select the printer required, together with page options.
2. Select the edit window that is to be printed. Note that it is not possible to print only part of the active window and should a hardcopy of part of a file be required it must first be copied into a temporary new window using the cut-and-paste facilities.
3. Select **File | Print** or Ctrl+P and complete the following dialogue box.



Print options dialogue box

6.1.9 Terminating RIDE

To terminate **RIDE** select **File | Exit**. If there are any open files, which have been modified but not saved, **RIDE** prompts for confirmation to save the files before exiting.

6.2 Editing text

When an editing window is open the text within it may be edited interactively using a cursor controlled by the mouse, keystrokes from the keyboard, the **Edit** menu and a pop-up menu.

6.2.1 Editor Options

The editor options, such as tab settings and font size, are saved within the project and may be modified via **Options | Project | Environment | Editor** from **RIDE**'s main screen. See section 5.2.3 for a full description

6.2.2 Moving to a line in the source file

To move to a specific line with the file :

1. Click the right button of the mouse to display the pop-up menu.
2. Select the command **Go To Line**, and the **Go To** dialog box appears.
3. Type the required line number.
4. Choose **OK**.

6.2.3 Using OEM characters

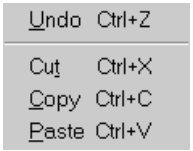
RIDE uses only the ANSI standard character set. However, it allows characters to be converted from OEM to ANSI standards when the source file is loaded from disk and to be converted from ANSI to OEM when the source file is saved to disk. To allow the conversion select **Options | Project | Environment | Editor**, check the box **Convert OEM to ANSI** and then select **OK**.

6.2.4 Using Tabulation characters

RIDE supports tab stops in a source file. To set the number of spaces a tab is equivalent to, select **Options | Project | Environment | Editor**, modify the value of **Tab Stops** and then select **OK**.

6.2.5 To transfer text

In an edit window, text may be highlighted using click and drag mouse operations. Once highlighted the text may be removed (or cut) and placed in the clipboard in a manner identical to many other Windows-based word-processors and text editors. Once in the clipboard the text may be pasted to another place within the current window or another edit window. It is also possible to duplicate the text using the copy option. The **Edit** window also reminds the user of the short-cut keys that are often used after a little practice.



Edit window options

Text may also be transferred to and from **RIDE** and other editors via the clipboard.

6.2.6 To undo an editing action

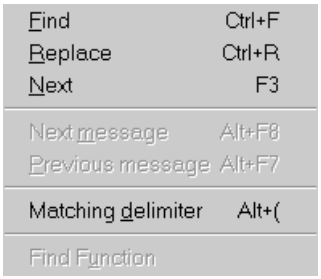
The **Edit | Undo** command (Ctrl+Z) allows unwanted changes to be restored. The maximum number of restorations is determined by **Options | Project | Environment | Editor| Undo number**.

Undo inserts any characters deleted, deletes any characters inserted, replaces any characters overwritten and moves the cursor back to a prior position. Block operation may also be undone. The Undo command will not change an option setting that affects more than one window or

reverse any toggle setting that has a global effect; for example, Ins/Ovr. This command is available only if an edit window is currently active and there is something to undo.

6.2.7 Finding and replacing text

It is possible to search for a given string by selecting **Search | Find** and completing the entries in the dialogue box.



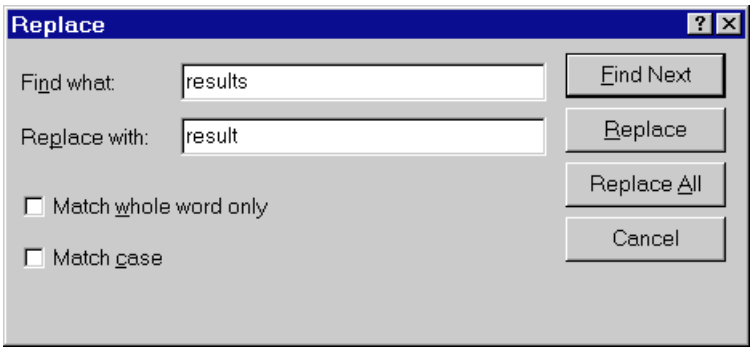
Search window Find dialogue box

The search takes place either up or down the file from the current cursor position and the editor retains the selected string between repeated uses of the uses of the **Find** command in any single session.

If a string is highlighted when the search option is selected it will automatically become the search string.

If the cursor is placed before an opening delimiter and **Search | Matching delimiter** is selected, the cursor will be moved so that it is placed immediately before the matching closing delimiter. Similarly, if it is placed before a closing delimiter this command will move the cursor so that it is placed immediately before the previous matching opening delimiter.

It is also possible to replace the next occurrence, or all subsequent occurrences, of one string with another. To do this select **Search | Replace** and completing the entries in the dialogue box.

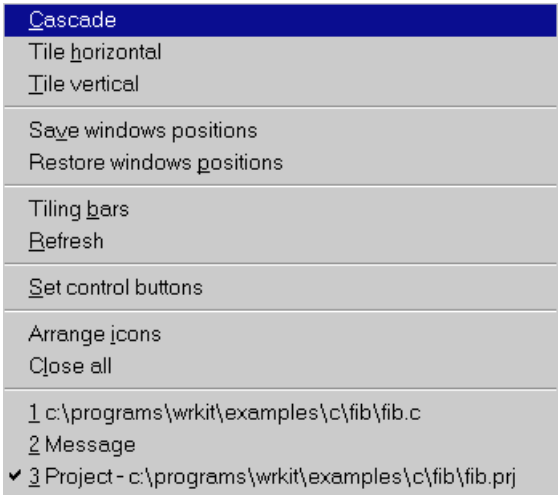


Replace Text dialogue box

6.3 Window management

Most of the windows in **RIDE** have all the standard window elements, such as scroll bars, Minimise and Maximise buttons, and a Control menu box.

The window management commands are selected from the window menu shown below.



Window options menu

Changes to the arrangement of the open windows apply to all but the project and message windows, which are always visible in the lower part of the screen. From the **Window** menu :

- **Cascade** : stacks all open windows and overlaps them so that each is approximately the same size but only part of each underlying window is visible.
- **Tile horizontal** : arranges the **RIDE** windows from top to bottom so that they cover the entire width of the desktop without overlapping one another. If there are more than three open windows, they are arranged in a manner that allows for more width than height.
- **Tile vertical** : is similar to Tile horizontal but windows are arranged from left to right.
- **Save windows position** : saves the positions of the windows.
- **Restore windows position** : restores the position of the windows.
- **Tile bars** : used as a splitter to organise the debug windows.
- **Refresh** : while executing a program, "refresh" will update the contents of all the debug windows. Moreover, it is possible to set a "Refresh point" to refresh automatically the windows each time the flag is met during execution.
- **Set control buttons** : brings up a window to select the control buttons, in various categories, which appear on the screen as short cuts to common commands.
- **Arrange icons** : rearranges the desktop icons in an iconized window.
- **Close all** : Closes all the opened windows
- There is a list of open windows at the bottom of the Window menu. When a window is active, a tick appears against its number and name. An inactive window may be selected by pressing the corresponding number of the keyboard, or by selecting the name with the mouse.

6.4 Commands summary

This table summarises the major commands within **RIDE**. Other icons exist and may be placed on the control bar via **Window | Set Control buttons**

Commands	Shortcut	Button	Description
New	Ctrl+N		Create a new source file
Open	Ctrl+O		Open an existing source file
Save	Ctrl+S		Save the file of the active window.
Exit	Alt+F4		Exit from RIDE
Undo	Ctrl+Z		Undo the last editor command.
Cut	Ctr+X		Remove the selected text and copies it to the clipboard
Copy	Ctrl+C		Copy the selected text to the Clipboard.
Paste	Ctrl+V		Paste from the clipboard in the position of the cursor.
Cut the current line	Ctrl+Y		Cut the editor line without copying the text into the clipboard.
Find			Find the specified text
Next	F3		Repeat the last Find or Replace command.
Next Message	Alt+F8		Find the next error position.
Previous Message	Alt+F7		Return to the last error.
Matching Bracket	Alt+(		Set the cursor position to the corresponding bracket.
Add File	Alt+Ins		Add a new file to the project.
Delete File	Alt+Del		Remove the selected file from the project.
Compile	Alt+F9		Compile or assemble the edited file.
Make	F9		Make the target of the current project.
Build all	Shift+F9		Rebuilds all the files in the project, regardless of whether they are out of date.
Tile			Tile open windows
Debug			activates the integrated debugger (or another tool)

7. INDEX