Getting Started with Red Suite 5

Version 5.1.0

29 January, 2013

Chapter 1. About this guide

1.1. Who should use it

This guide is intended as an introduction to using Red Suite 5 for creating Microcontroller (MCU) application software.

This guide assumes that you have some knowledge of MCUs and software development for embedded systems.

Chapter 2. Red Suite 5

2.1. Overview

Red Suite 5 is a fully featured software development environment for ARM- based MCUs, and includes all the tools necessary to develop high quality embedded software applications in a timely and cost effective fashion.

The Red Suite 5 IDE is based on the Eclipse IDE and features many ease-of-use and MCU specific enhancements. Red Suite 5 also includes the industry standard ARM GNU tools enabling professional quality tools at low cost. The fully featured debugger supports both SWD and JTAG debugging, and features direct download to on-chip flash.

Red Suite 5 supports a wide range for MCUs from Freescale, NXP, Silicon Labs, STMicroelectronics and Texas Instruments.

2.2. New features

Red Suite 5 builds upon the functionality provided in the previous generation Red Suite 4.

New functionality contained in Red Suite 5 includes:

- The IDE is now based on the Eclipse “Juno” release and CDT 8.1.
- The compilation tools are now based on GCC 4.6.2
- Red Trace enhancements
  - Support for Instruction Trace for Cortex-M3/M4 based MCUs implementing an Embedded Trace Buffer (ETB).
  - Support for Instruction trace for Cortex-M0+ based systems implementing a Micro Trace Buffer (MTB).
- Additional MCU Support
  - Support for Freescale Kinetis MCUs extended to include Kinetis L parts (Cortex-M0+).
  - Support for NXP MCUs extended to include LPC11xxLV parts (Cortex-M0).
  - Support for STMicroelectronics MCUs extended to include STM32F0 parts (Cortex-M0).
- Updated Redlib C library

  - Cortex-M4 specific library variants now provided, rather than just relying on Cortex-M3 variant. Should lead to some performance improvements, particularly when (single precision) floating point is being used.
  - `printf` family functions no longer pull in `ctype` constant data, saving ~400 bytes of flash space for applications using any of these functions.
- Red State enhanced to support multiple state machines in a single project
- Additional Debug probe support
  - Support for direct debug connection to Code Red’s RDB4078 board.
  - Support for direct debug connection to Freescale FreedomKL25Z board.

Chapter 3. Installation and Licensing

3.1. Host PC Hardware Requirements

Before installation you should make sure your development host PC meets the following requirements (note that Red Suite NXP Editions are only supported on Windows platforms):

- A standard x86 PC with 2GB RAM minimum (4GB recommended) and 500MB+ of available disk space, running one of the following operating systems:
  - Microsoft® Windows XP (SP2 or greater)
  - Microsoft® Windows Vista
  - Microsoft® Windows 7
  - Microsoft® Windows 8
  - Linux – Ubuntu 9 thru 12
  - Linux – Fedora 14 thru 17

Both 32-bit and 64-bit systems are supported. Note that Red Suite may install and run on other Linux distributions. However, only the distributions listed above have been tested. We have no plans to officially support other distributions at this time.

- An x86 Apple Macintosh with 2GB RAM minimum (4GB recommended) and 750MB+ of available disk space, running one of the following operating systems:
  - Mac OS X version 10.7.5 or later
  - Mac OS X version 10.8.2 or later

A screen resolution of 1024x768 minimum is recommended.

An internet connection is **required** to request, install and activate license keys. When using the product, an internet connection is used to update the product using the online update mechanism and to download new examples.

Supported debug probes are

- Code Red’s Red Probe, Red Probe+, RDB1768 and RDB4078
- Freescale’s FreedomKL25Z board
- NXP’s LPC-Link

- Silabs USB debug adapter
- TI’s ICDI

3.2. Installation

3.2.1. Windows

Red Suite is installed into a single directory, of your choice. Unlike many software packages, Red Suite does not install or use any keys in the Windows Registry, or use or modify any environment variables (including PATH), resulting in a very clean installation that does not interfere with anything else on your PC. Should you wish to use the command line tools, a command file is provided to set up the path for the local command window.

3.2.2. Linux

The Red Suite installer is supplied as a GUI executable that runs and installs the required components. The installer requires root privileges, although once installed, no special privileges are required to run Red Suite. The installer will request a super-user password when started. Once installation has completed, we strongly recommend that your system is restarted — if you do not do this then some areas of the tools may not function correctly

On Fedora, the installer must be started from the command line, but will switch to GUI mode once the super-user password has been entered. On Ubuntu, the super-user password is requested in GUI mode.

Ubuntu 9 thru 12 For 64-bit versions of this distribution, the 32-bit compatibility components must be installed. Note that without these components installed, the installation program will not run. To install these components from the command line, the following command should be executed:

```
sudo apt-get install linux32 ia32-libs
```

Ubuntu 11.04 (Natty Narwal) and later If using the new Unity interface, there is an issue preventing some menu items from displaying in the Red Suite IDE (this does not affect the ‘Classic’ interface). To workaround this problem, create a shell script with the following content, and start Red Suite by running the below script...

```
#!/bin/bash
export UBUNTU_MENUPROXY=0
/usr/local/<redsuite_install_dir>/redsuite/redsuite
```

Fedora 12 and greater If SELINUX is used, it must be set to “permissive” mode to allow Red Suite to run

3.2.3. Mac OS X

The Red Suite installer is supplied as a Mac OS X `.pkg` installer file. Double click on the installer to install Red Suite into a subfolder of your Applications folder.

To start Red Suite, use the Mac OS X Launchpad. Alternatively click the **Open redsuite** icon in the `/Applications/redsuite_version_ folder` or run **redsuite.app** that can be found in the `redsuite` subfolder of the main Red Suite installation directory within `/Applications`.

3.2.4. Running under virtual machines

Although it is possible to install Red Suite within a virtual machine environment, we do not support such usage. In particular you may encounter timeout related issues when carrying out debug operations.

3.3. License Installation

All components of Red Suite are installed, but some functionality may be restricted by the currently installed license activation. Through registration and activating of your software, Red Suite will give you full development environment when connected to your target through a Red Probe+, Red Probe or LPC-Link debug interface, or connected directly to a Code Red or other supported development board.

3.3.1. Unregistered (UNREGISTERED)

Initially Red Suite is supplied with an Unregistered (UNREGISTERED) license. All features of the product may be used, although some functionality is restricted, including the size of debug image (limited to 8Kbytes), and Red Trace functionality is disabled.

3.3.2. Evaluation Licence (EVALUATION)

An Evaluation (EVALUATION) license may be obtained, free of charge, by registering the product through the **Help->Product Activation** menu of the Red Suite IDE. This license gives a complete development environment for a period of up to 60-days. Some code size and other limitations may remain, including only allowing a single run of the Red Trace feature per debug session. Once registered, an Activation Code will be emailed to the address supplied.

Important Notes:

- An Evaluation license is for Evaluation of the Red Suite product and is not a license to use it in any commercial development.
- Red Suite (NXP Edition) does not support Evaluation licenses. The Evaluation product for the NXP Edition is Red Suite Full edition.

3.3.3. Commercial Use License (FULL)

A Full (FULL) license may be purchased from the Code Red Technologies web site. http://www.code-red-tech.com/store.html

- For Red Suite, a full license removes all restrictions on any target.
- For Red Suite (NXP Edition) a full license permits development of applications up to the size permitted by the license (256k or 512k).

3.3.4. Activating your EVALUATION or FULL license

You will require an Internet connection to request and activate your license. To activate your Evaluation, or Full license, enter the Activation Code (as received via email), into the dialog opened via the **Help->Product Activation->Activate license** menu. You will be required to enter a password that will be associated with the license. Please make note of this password as it can be used to manage your license, including preparation to move it to another machine.

3.3.5. Use of your email address

The email address on the activation and reactivation dialogs is required, so that we may send you your activation code. The email address will not be sold or provided to any third party. You may unsubscribe from any emails that we may send you.

3.3.6. More information on licensing

For more information on activation and other areas of licensing, please see the FAQs at: http://support.code-red-tech.com/CodeRedWiki/CodeRedFAQ#ToolsLicensing

Chapter 4. Red Suite Overview

4.1. Documentation and Help

Red Suite is based on the Eclipse IDE framework and many of the core features are described well in generic Eclipse documentation and in the help files to be found on Red Suite's **Help -> Help Contents** menu. This also provides access to Getting Started guides for Red Suite (this document), Red Trace and Red State, as well as the documentation for the compiler, linker and other underlying tools.

Further documentation and FAQs are also available on the Code Red Technologies support site: http://support.code-red-tech.com

To request assistance on using Red Suite, visit: http://code-red-tech.com/support.php

4.2. Workspaces

When you first launch Red Suite, you will be asked to select a Workspace, as per Figure 4.1.

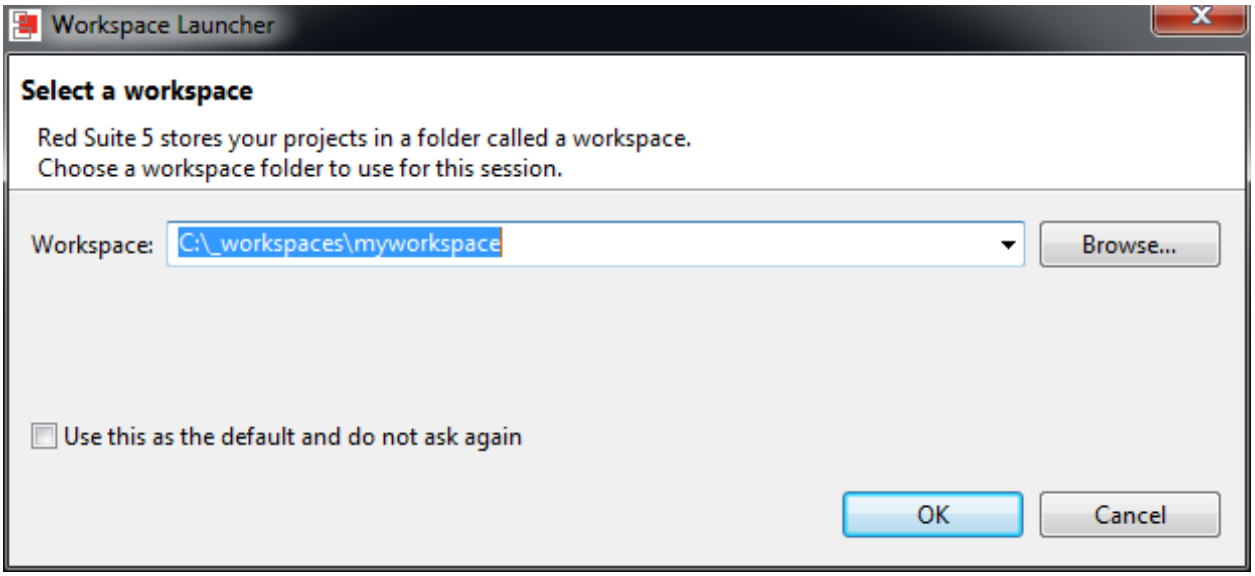


Figure 4.1. Workspace selection.

A workspace is simply a directory which is used to store the projects that you are currently working on. Each workspace can contain multiple projects, and you can have multiple workspaces on your computer. Red Suite can only have a single workspace open at a time, although it is possible to run multiple instances of Red Suite in parallel — with each instance accessing a different workspace.

If you tick the **Use this as the default and do not ask again** option, then Red Suite will always start up with the chosen workspace opened. Otherwise you will always be prompted to choose a workspace.

It is also possible to change workspace whilst running Red Suite, using the **File -> Switch Workspace** option.

4.3. Perspectives and Views

The overall layout of the main Red Suite window is known as a Perspective. Within each Perspective are many sub-windows, called Views. A View displays a particular set of data in the Red Suite environment. For example this data might be source code, hex dumps, disassembly or memory contents. Views can be opened, moved, docked, and closed, and the layout of the currently displayed Views can be saved and restored.

Typically, Red Suite operates in a ‘dual perspective’ mode. The **C/C++ Perspective** is used for developing and navigating around your code. The **Debug Perspective** is used when debugging your application.

Red Suite also supports the single **Develop Perspective** under which both the code development and debug sessions operate as shown in Figure 4.5. The single perspective simplifies the Eclipse environment, but at the cost of reducing the amount of information displayed on screen.

You can manually switch between Perspectives using the Perspective icons in the top right of the Red Suite window, as per Figure 4.2. In addition when you launch a debug session from the **C/C++ Perspective**, the IDE will normally automatically switch you to the **Debug Perspective**.

All Views in a Perspective can also be rearranged to match your specific requirements by dragging and dropping. If a View is accidentally closed, it can be restored by selecting it from the **Window -> Show View** dialog.

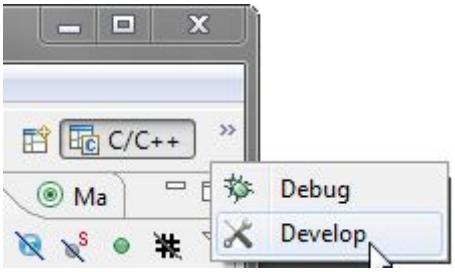


Figure 4.2. Perspective selection.

4.4. Major components of the C/C++ Perspective

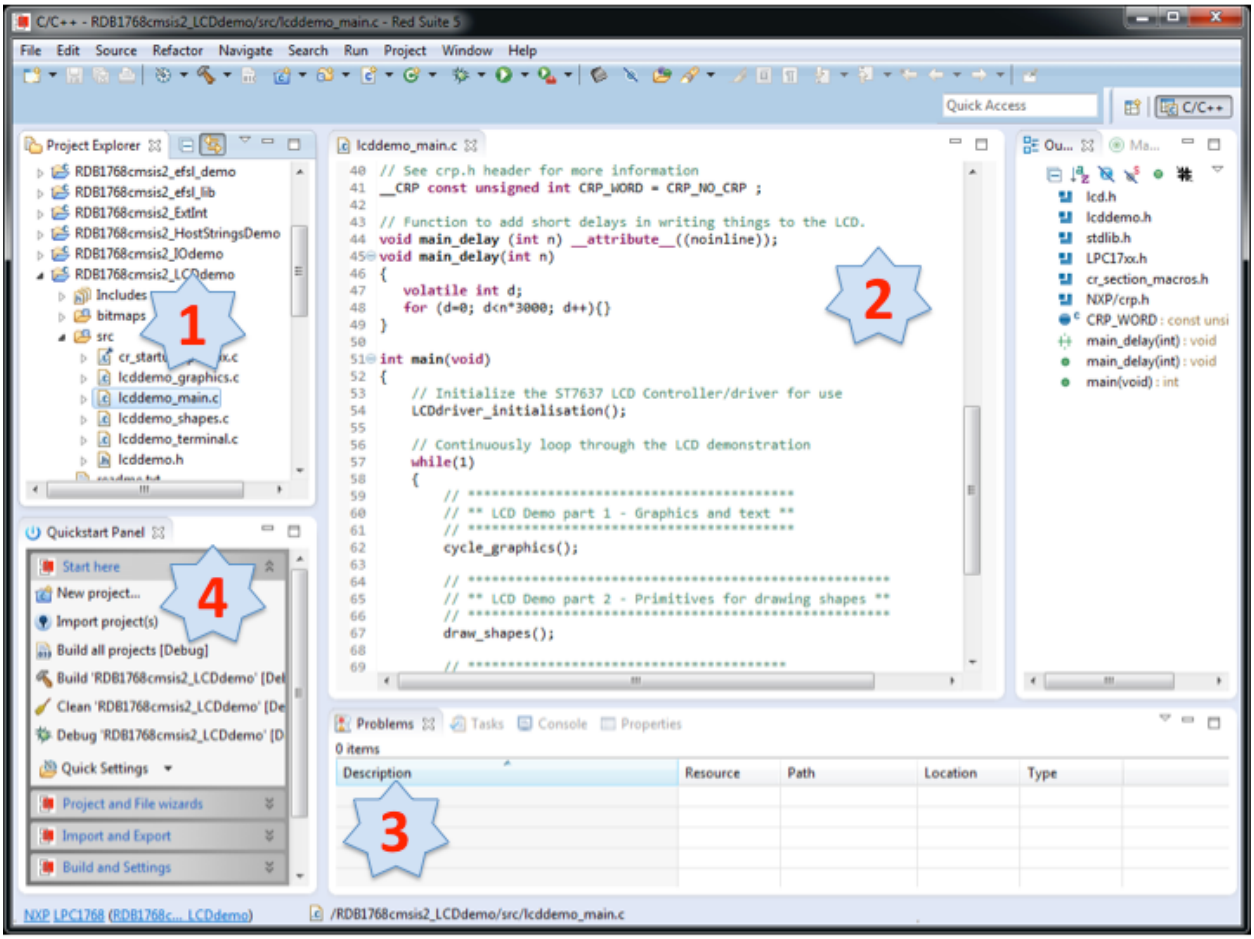


Figure 4.3. C/C++ Perspective

1. Project Explorer View

   - The **Project Explorer** gives you a view of all the projects in your current ‘Workspace’.

2. Editor

   - On the upper right is the editor which allows modification and saving of source code as well as setting breakpoints in debug mode.

3. Console and Problems Views

   - On the lower right are the **Console** and **Problems** Views. The **Console** View displays status information on compiling and debugging, as well as program output. The **Problem** View (available by changing tabs) shows all compiler errors and allows easy navigation to the error location in the **Editor** View.

4. Quick Start View

   - On the lower left, the **Quick Start** view has fast links to commonly used features. This is the best place to go to find options such as **New project**, **Build**, **Debug**, and **Import**.

4.5. Major components of the Debug Perspective

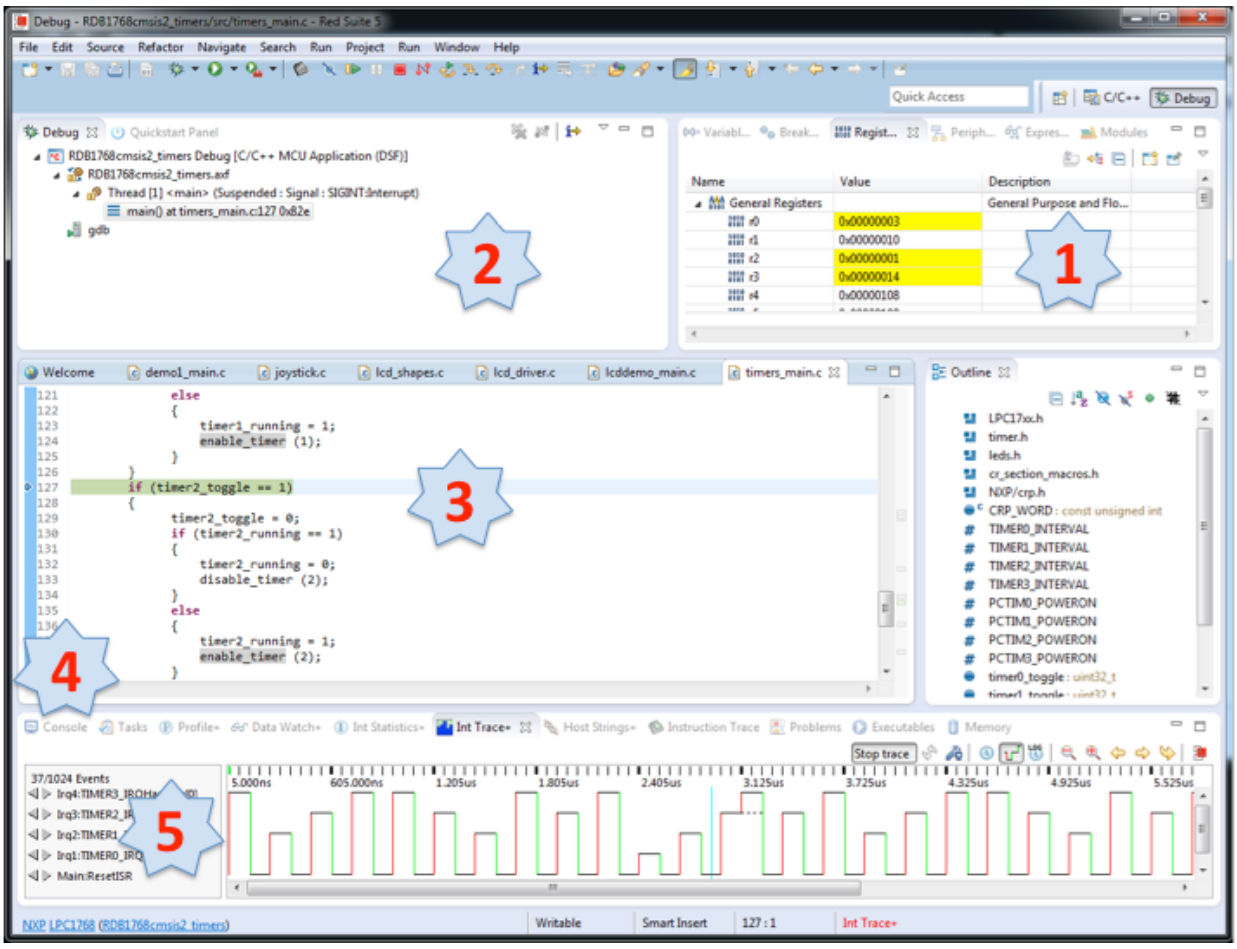


Figure 4.4. Debug Perspective

1. Registers / Variables / Breakpoints / Expressions Views

   - This shows all of the registers in the processor core. Registers that have changed from step to step are highlighted in yellow.

- Sitting in parallel to the ‘Registers’ view, the **Variables** view allows you to see the values of local variables.

- Sitting in parallel to the ‘Registers’ view, the **Breakpoints** view allows you to see and modify currently set breakpoints.

- Sitting in parallel to the ‘Registers’ view, the **Expressions** view allows you to add global variables and other expressions so that you can see and modify their values.

2. Debug View

   - This shows you the stack trace. In the ‘stopped’ state, you can click on any particular function and inspect its local variables in the Variables tab (parallel to the Registers View). Debug controls can be found on the global toolbar (at the top of the screen).

3. Editor

   - In here you will see the code you are executing and can step from line to line. By pressing the `'i->'` icon at the top of the Debug view, you can switch to stepping by assembly instruction. Clicking in the left margin will set and delete breakpoints.

4. Console View

   - On the lower left is the Console View. During debugging, the Console View provides various pieces of status information as well as semihosted program output.

5. Red Trace

   - Located in parallel with the Console View, are the various views that make up the Red Trace functionality of Red Suite. The Red Trace views allow you to gather and display runtime information using the SWV technology that is part of Cortex-M3/M4 based parts. The example here shows the Interrupt Trace view. In addition for some MCUs you can also view instruction trace data downloaded from the MCU’s Embedded Trace Buffer (ETB) or Micro Trace Buffer (MTB). For more information on Red Trace functionality, please see the **Getting Started with Red Trace** document.

4.6. Major components of the Develop Perspective

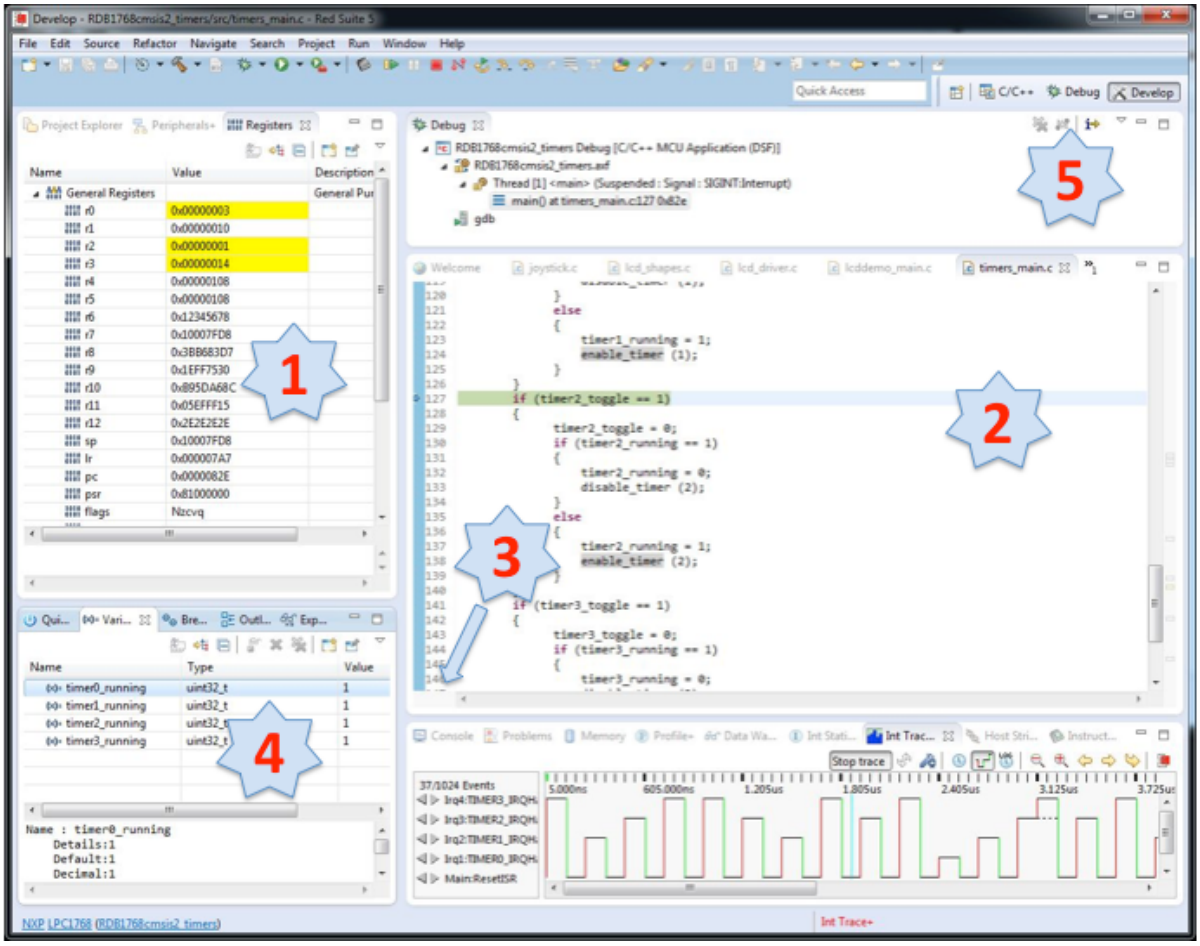


Figure 4.5. Develop Perspective (whilst debugging)

1. Project Explorer / Peripherals / Registers Views

   - The **Project Explorer** gives you a view of all the projects in your current ‘Workspace’.
   - When debugging the **Peripherals** view allow you to display the registers within Peripherals
   - When debugging the **Registers** view allow you to display the registers within the CPU of your MCU.

2. Editor

   - On the upper right is the editor which allows modification and saving of source code. When debugging, it is here that you will see the code you are executing and can step from line to line. By pressing the `'i->'` icon at the top of the Debug view, you can switch to stepping by assembly instruction. Clicking in the left margin will set and delete breakpoints.

3. Console / Problems / Red Trace Views

- On the lower right are the Console and Problems Views. The Console View displays status information on compiling and debugging, as well as semihosted program output. The Problem View (available by changing tabs) shows all compiler errors and will easy navigation to the error location in the Editor View.

- Located in parallel with the Console View, are the various views that make up the Red Trace functionality of Red Suite. The Red Trace views allow you to gather and display runtime information using the SWV technology that is part of Cortex-M3/M4 based parts. The example here shows the Interrupt Trace view. In addition for some MCUs you can also view instruction trace data downloaded from the MCU's Embedded Trace Buffer (ETB) or Micro Trace Buffer (MTB). For more information on Red Trace functionality, please see the **Getting Started with Red Trace** document.

4. Quick Start / Variables / Breakpoints / Expressions Views

   - On the lower left of the window, the **Quickstart Panel** has fast links to commonly used features. This is the best place to go to find options such as Build, Debug, and Import

   - Core Register View

   - Sitting in parallel to the 'Quickstart' view, the **Variable** view allows you to see the values of local variables.

   - Sitting in parallel to the 'Quickstart' view, the **Breakpoint** view allows you to see and modify currently set breakpoints.

   - Sitting in parallel to the 'Quickstart' view, the **Expressions** view allows you to add global variables and other expressions so that you can see and modify their values.

5. Debug View

   - The Debug view appears when you are debugging your application. This shows you the stack trace. In the 'stopped' state, you can click on any particular function and inspect its local variables in the Variables tab (which is located parallel to the **Quickstart Panel**). .

Chapter 5. Getting started with an Example project

The **Quickstart Panel** provides rapid access to the most commonly used features of Red Suite. Using the **Quickstart Panel**, you can quickly import example projects, create new projects, build projects and debug projects.

5.1. Importing an Example project

On the **Quickstart Panel**, click on the 'Start Here' sub-panel, and click on Import project(s).

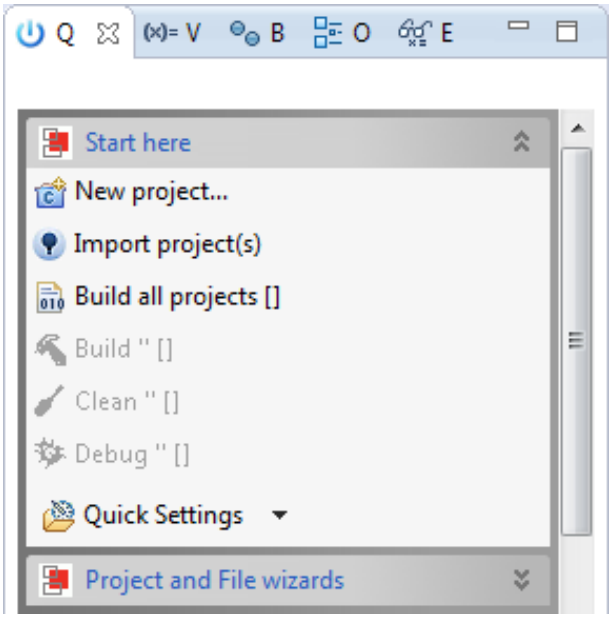


Figure 5.1. Import project(s)

As per Figure 5.2 from the first page of the wizard, you can

- Browse to locate Examples stored in zip archive files on your local system.
- Browse to locate projects stored in directory form on your local system (for example you can use this to import projects from a different workspace into the current workspace).
- Browse the web to locate and download examples onto your local system. Browse the web will automatically open a Web Browser on the Code Red support website containing examples suitable for your installation.

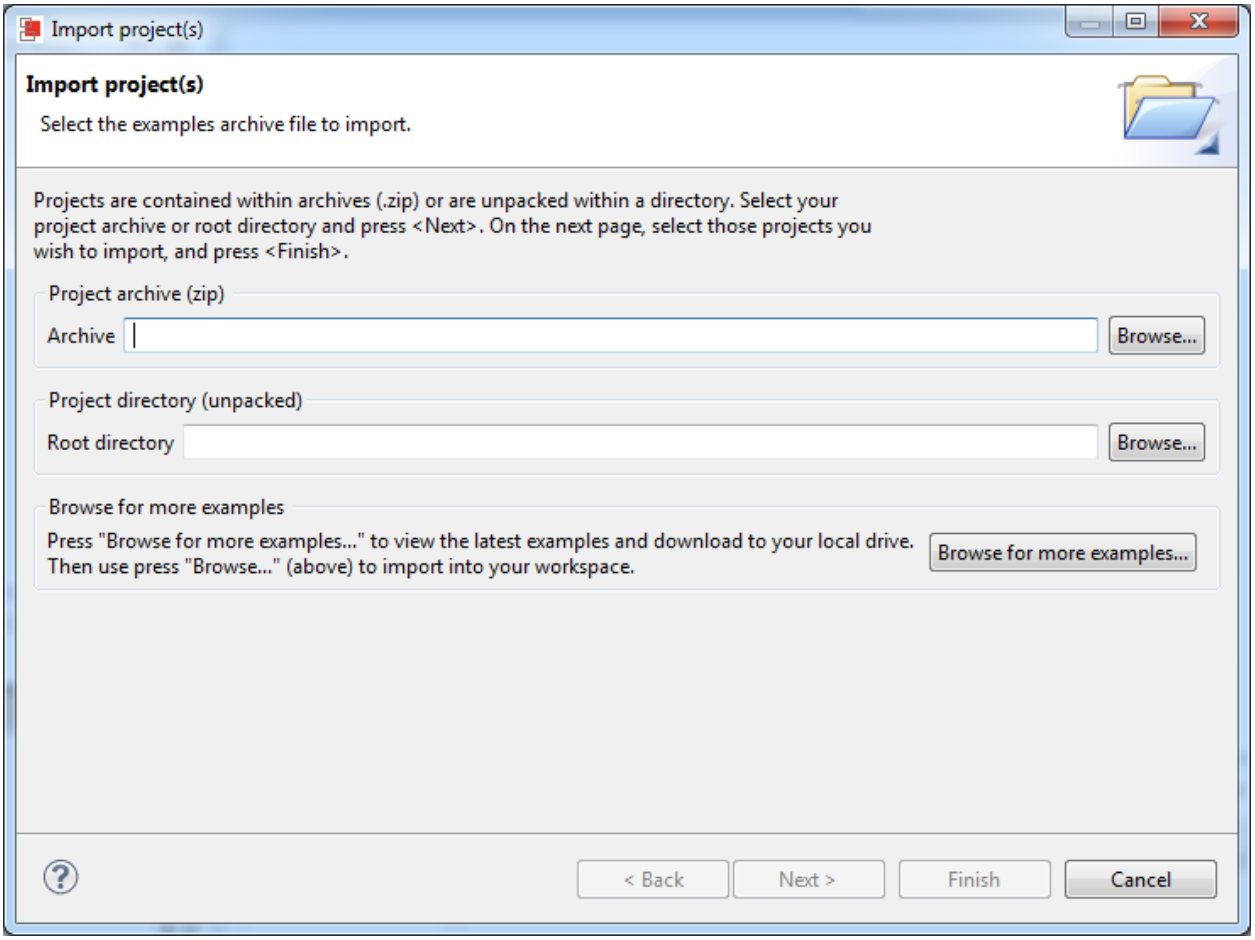


Figure 5.2. Import Examples

5.1.1. Importing Examples for the RDB1768 Development board

Click on **Browse** and locate the NXP LPC17xx examples directory within your Red Suite installation.

Once in Examples directory, select the `RDB1768cmsis2.zip` archive, click **Open** and then **Next**. You will then be presented with a list of projects within the archive as shown in Figure 5.3.

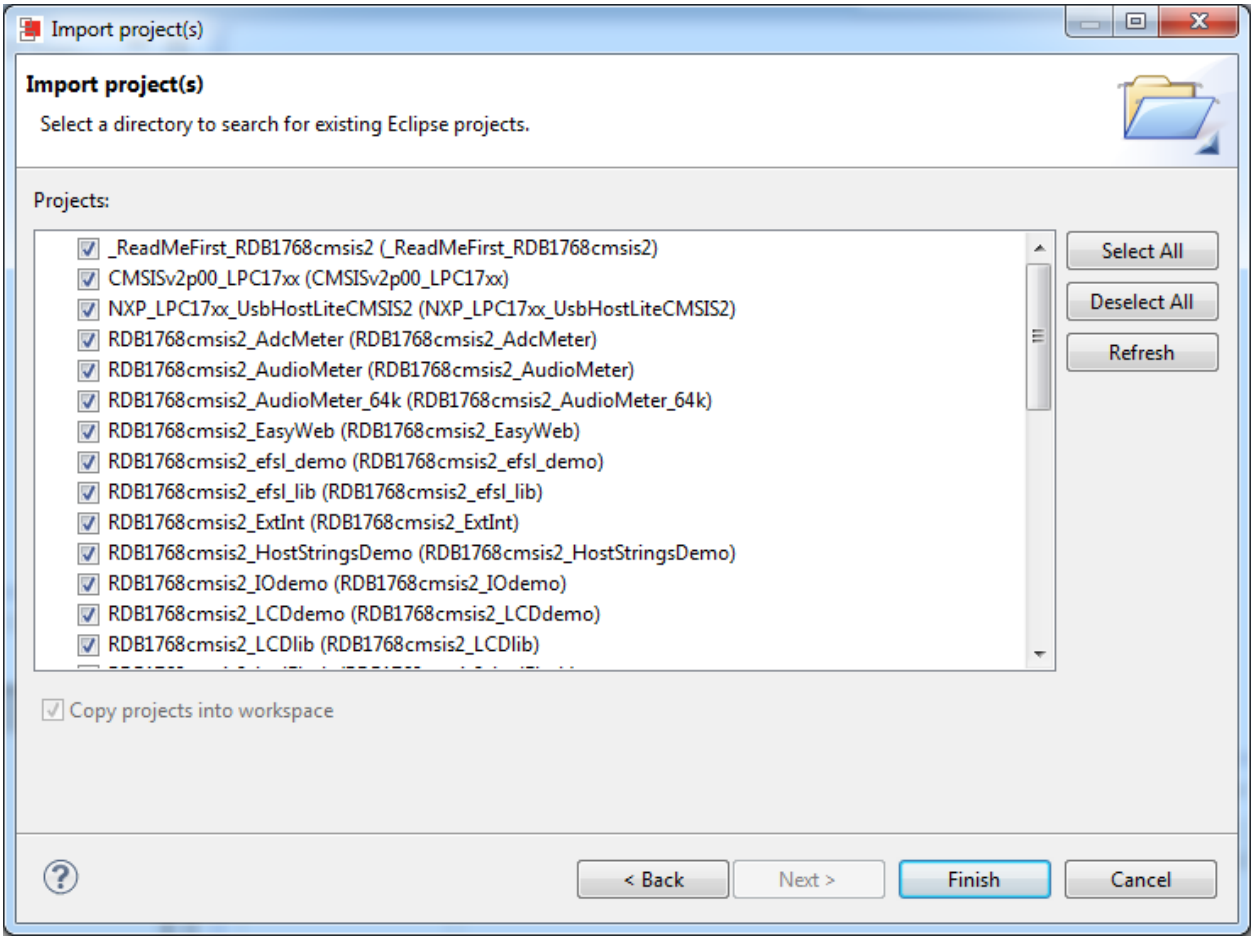


Figure 5.3. Select projects to import

Select the projects you want to import (if in doubt, leave all projects selected) and then click **Finish**. The examples will be imported Red Suite.

5.2. Building the Examples

Building the examples is a simple case of using the **Quickstart Panel** to Build all projects. Alternatively a single project can be selected and built.

5.3. Debugging the LCDdemo example

This description shows how to run the `RDB1768cmsis2_LCD_demo` application on an RDB1768 development board. When debug is started, the program is automatically downloaded to the target and is programmed into FLASH memory, a breakpoint is set on the first instruction in `main()` the application is started (by simulating a processor reset) and code executed until the first breakpoint is hit.

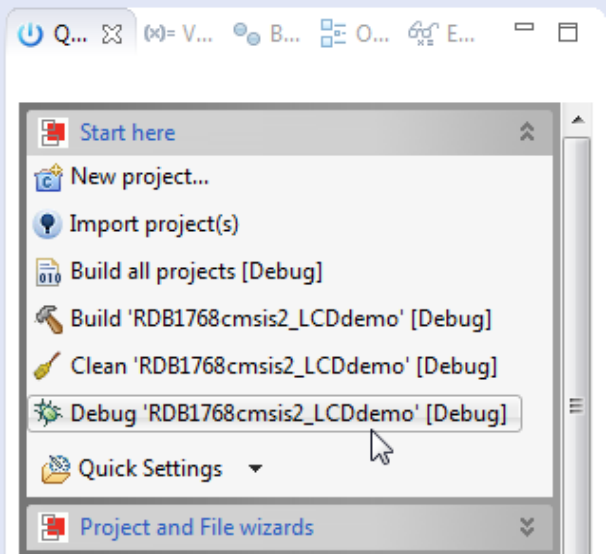


Figure 5.4. Launch debug session

To start debugging `RDB1768cmsis2_LCDdemo` on your target, simply highlight the project in the Project Explorer and then in the **Quickstart Panel** click on **Start Here** and select **Debug project RDB1768cmsis2_LCDdemo**, as per Figure 5.4.

Red Suite will first build and then start debugging the application. Click on **OK** to continue with the download and debug of the 'Debug' build of your project.

If you are using the **C/C++ Perspective** for project creation, then the first time you debug a project in a particular workspace, you will be asked to **Confirm Perspective Switch**. Tick the box saying **Remember my decision** and then **Yes**, as per Figure 5.5. You will then be presented with the **Debug Perspective** and will have run control over the application code running on your target.

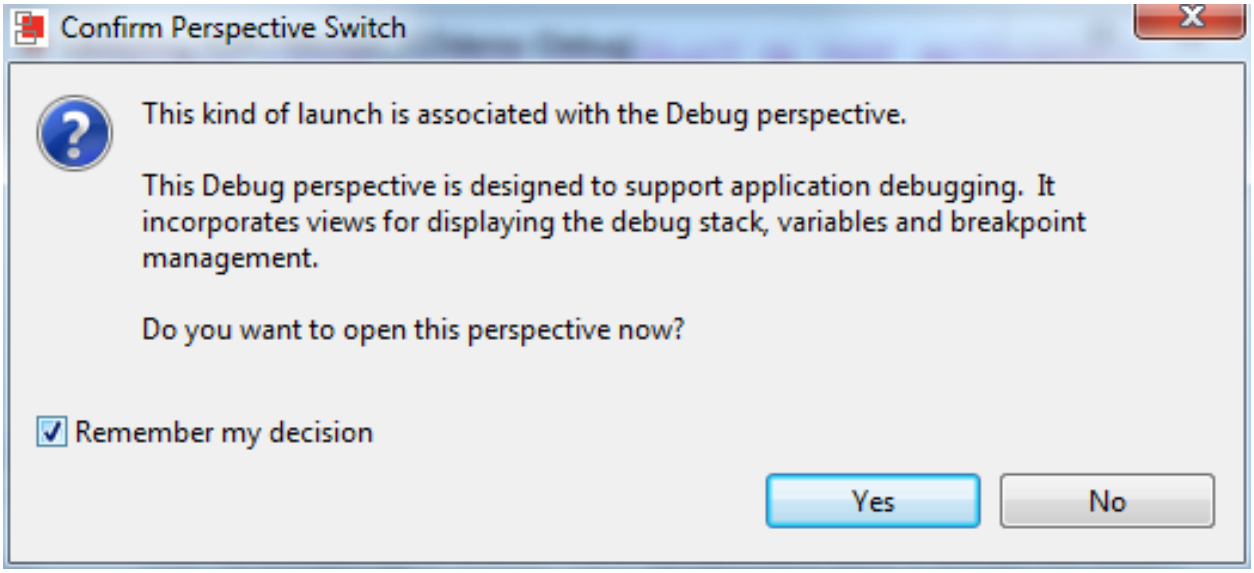


Figure 5.5. Perspective Switch

5.3.1. Controlling execution

Program execution can be controlled using the common debug control buttons on the global toolbar. The call stack is then shown in the Debug View.

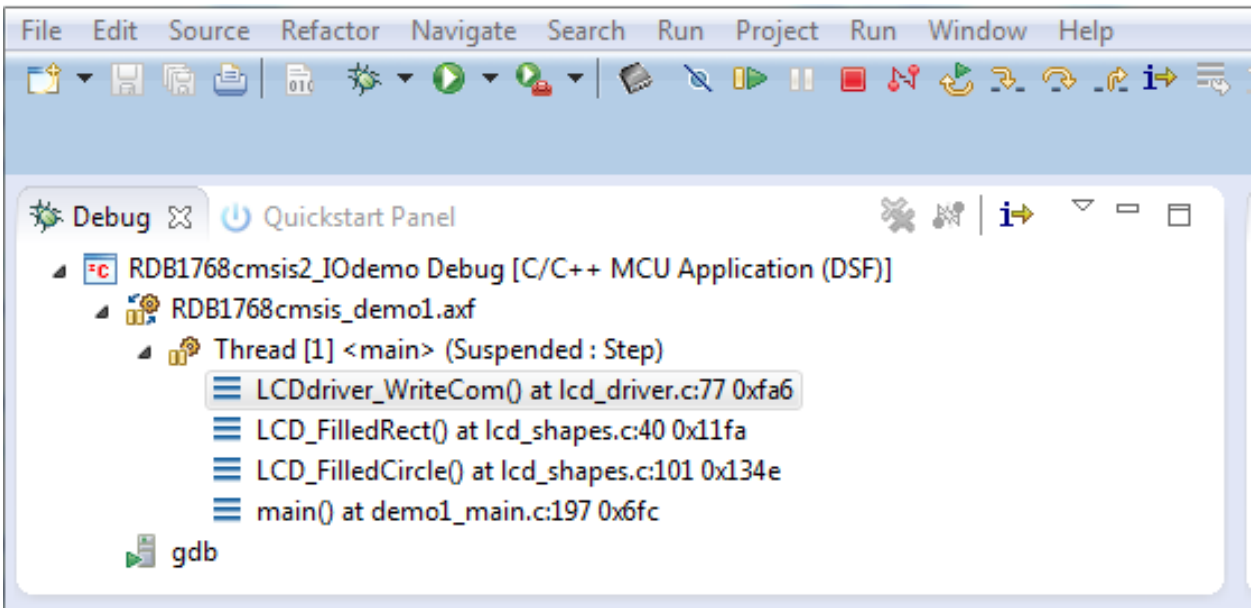


Figure 5.6. Debug Controls

Table 5.1. Program execution controls

Button	Description	Short-cut
	Restart program execution (from reset)	
	Run/Resume the program.	F8

Button	Description	Short-cut
	Step Over C/C++ line.	F6
	Step into a function.	F5
	Return from a function	F7
	Stop the debugger.	Ctrl + F2
	Pause Execution of the running program.	
	Show disassembled instructions.	

Note – In Red Suite 4 and earlier, the common debug control commands were found on the Debug View's local toolbar, rather than on the global toolbar. Moving them to the global toolbar allows the Debug View to be minimized when not needed – allowing more space on screen for other views to be displayed.

Setting a breakpoint

To set a breakpoint, simply double-click on the margin area of the line you wish to set a breakpoint on (before the line number).

Restart application

If you hit a breakpoint or pause execution and want to restart execution of the application from the beginning again, then you can do this using the **Restart** button.

Stopping debugging

To stop debugging just press the **Stop** button.

If you are debugging using the **Debug Perspective**, then to switch back to **C/C++ Perspective**, just click on the **C/C++** tab in the upper right area of Red Suite (as per Figure 4.2).

Chapter 6. Creating Projects using the Wizards

Red Suite includes many project templates to allow the rapid creations of correctly configured projects for specific MCUs.

6.1. Creating a project using the wizard

Click on the **New project...** option in the Start here tab of the **Quickstart Panel** to open up the Project Creation Wizard.

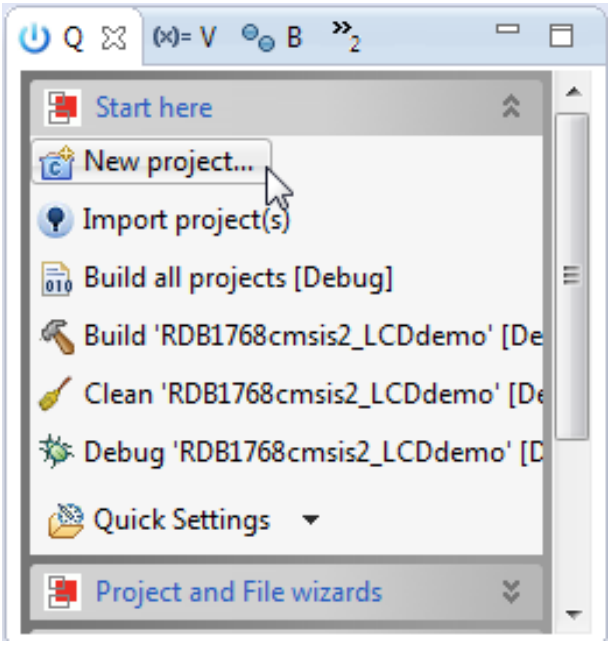


Figure 6.1. New Project

Now click on the “expander” (small triangular icon) next to the name of the MCU vendor you are using (for example NXP or TI), then select the MCU family that you wish to create a new project for, as per Figure 6.2. Note that in some cases, a number of families of MCUs are grouped together at a top level – just open up the appropriate “expander” to get to the family that you require. In this example the top level group LPC4x is used to hold all of NXP’s LPC4000 families (LPC407x_8x and LPC43xx).

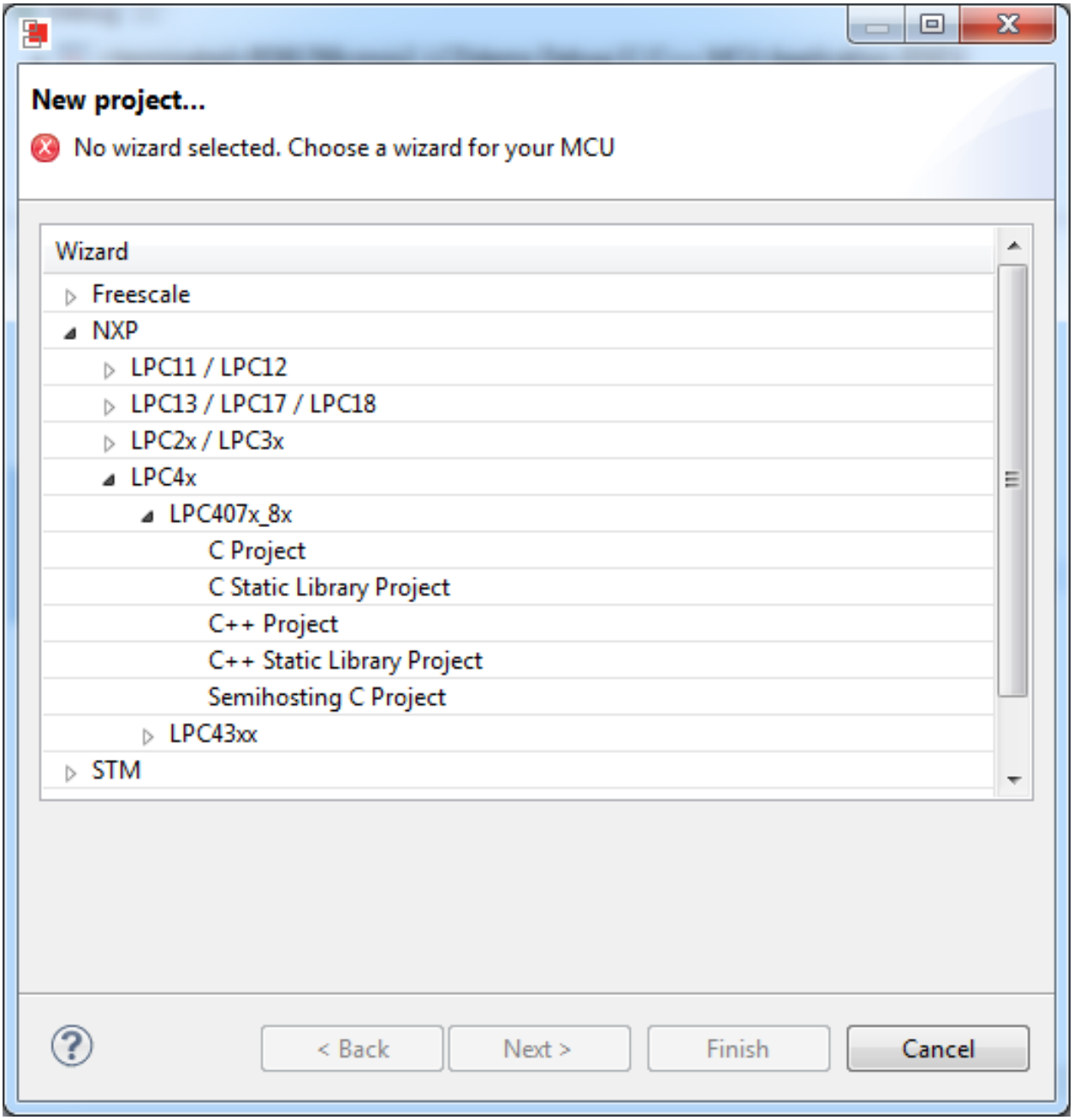


Figure 6.2. New Project: wizard selection

You can now select the type of project that you wish to create. Most MCU families will provide a default set of five wizards, consisting of:

C Project

- Creates a simple C project, with the `main()` routine consisting of an infinite `while(1)` loop which increments a counter.

C++ Project

- Creates a simple C++ project, with the `main()` routine consisting of an infinite `while(1)` loop which increments a counter.

Semihosting C Project

- Creates a simple “Hello World” project, with the `main()` routine containing a `printf()` call, which will cause the text to be displayed within the Console View of Red Suite. This is implemented using Red Suite’s “semihosting” functionality. For more details, please see the FAQ at http://support.code-red-tech.com/CodeRedWiki/UsingPrintf

C Static Library Project

- Creates a simple static library project, containing a source directory, and optionally a directory to contain include files. The project will also contain a “liblinks.xml” file, which can be used by the smart update wizard on the context sensitive menu to create links from application projects to this library project. For more details, please see the FAQ at http://support.code-red-tech.com/CodeRedWiki/StaticLibrarySupportv4

C++ Static Library Project

- Creates a simple (C++) static library project, as per the C Static Library Project wizard, but with the tools set up to build C++ rather than C code.

Some MCU families may also provide additional project wizards, for example to create a FreeRTOS project, or a project that uses the StellarisWare driver library (TI/Luminary parts).

Once you have selected the appropriate project wizard, you will be able to enter the name of your new project.

Then you will need to select the actual MCU that you will be targeting for this project. It is important to ensure that the MCU you select matches the MCU that you will be running your application on. This makes sure that appropriate compiler and linker options are used for the build, as well as correctly setting up the debug connection.

Finally you will be presented with one or more “Options” pages that provide the ability to set a number of project specific options. The options presented will depend upon which MCU you are targeting and the wizard you selected, and may also change between versions of Red Suite. Figure 6.3 gives an example options page for an LPC17xx project. Note that if you have any doubts over any of the options, then we would normally recommend leaving them set to their default value.

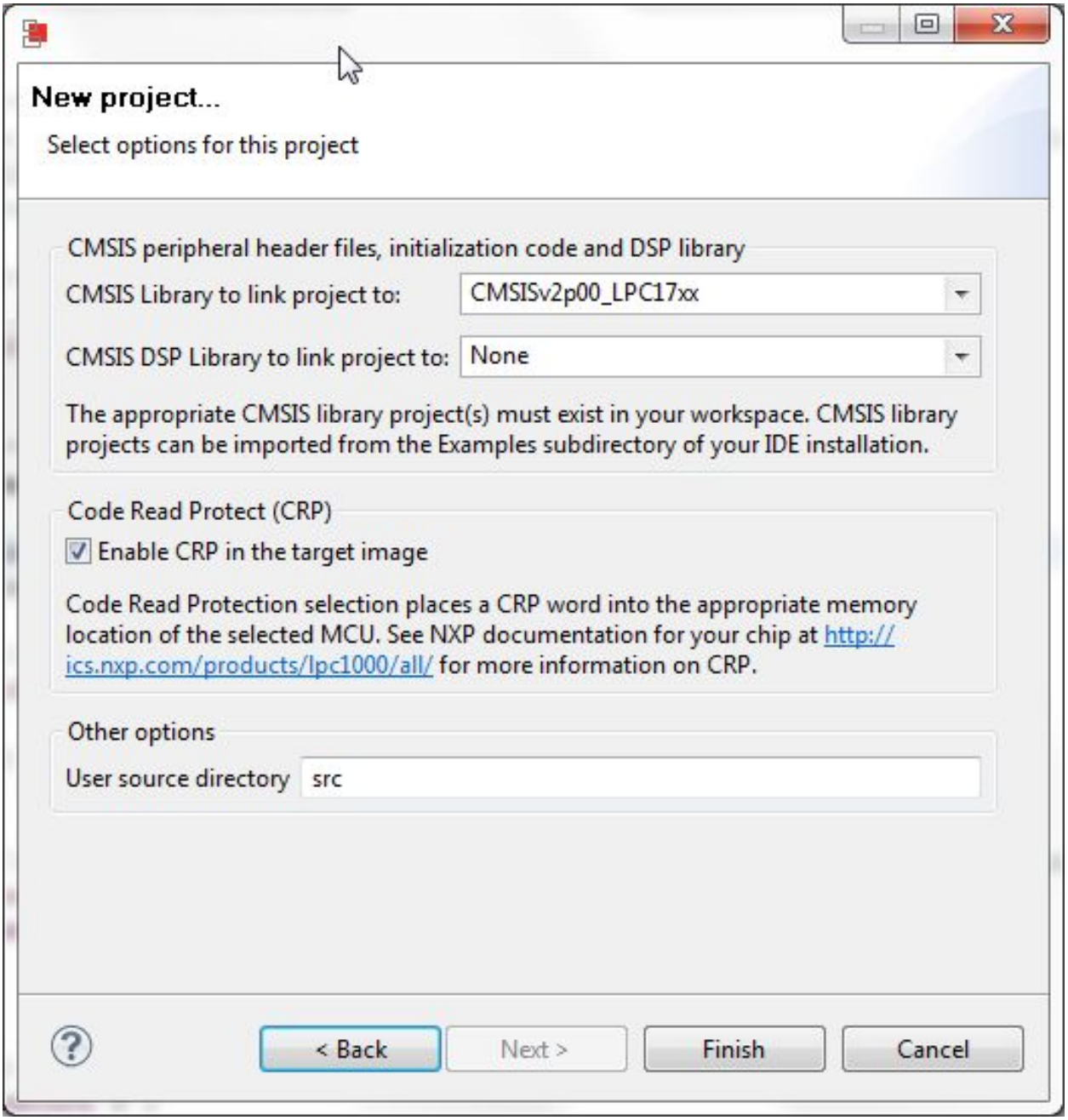


Figure 6.3. LPC17xx project wizard options page

The following sections detail some of the options that you may see.

6.1.1. CMSIS selection

The Cortex Microcontroller Software Interface Standard provides a defined way of accessing MCU peripheral registers, as well as code for initializing an MCU and accessing various aspects of functionality of the Cortex CPU itself. Red Suite supports the use of versions 1.3 and 2.0 of CMSIS for many MCUs (though only CMSIS 2.0 library projects are provided for some MCUs).

This option allows you to select which (if any) CMSIS library you want to link to from the project that you are creating. Note that the appropriate CMSIS library project must have been imported into

the workspace before starting the wizard, otherwise an error will be given when the wizard attempts to create your project. CMSIS library projects can be found in the Examples directory of your Red Suite installation.

For more information on CMSIS and its support in Red Suite, please see that FAQ http://support.code-red-tech.com/CodeRedWiki/Support4CMSIS .

6.1.2. CMSIS DSP library selection

As well as providing peripheral register access and startup code, CMSIS 2.0 also provides a DSP library. You can choose to make use of this DSP library separately from the main CMSIS selection. Note that Red Suite provides a separate library project for the DSP library to the main CMSIS 2.0 library project. This DSP library project is provided in pre-built form for Cortex-M3 and Cortex-M4 parts, though a source version of the DSP library is also provided within the Red Suite Examples.

6.1.3. Code Read Protect

NXP's Cortex and ARM7 based MCUs provide a mechanism to prevent certain types of access to the flash memory by external tools when a specific memory location in flash contains a specific value. Red Suite provides support for setting this memory location. For more details see the FAQ at http://support.code-red-tech.com/CodeRedWiki/CodeReadProtect .

6.1.4. Enable use of floating point hardware

Certain MCU's may include a hardware floating point unit (for example NXP LPC32xx and LPC43xx parts). This option will set appropriate build options so that code is built to use the hardware floating point unit and will also cause start up code to enable the unit to be included.

6.1.5. Enable use of `Romdivide` library

Certain MCU's, such as NXP LPC12xx, include optimized code in ROM to carry out divide operations. This option enables the use of these Romdivide library functions. For more details see that FAQ at http://support.code-red-tech.com/CodeRedWiki/RomDivide .

6.1.6. Disable Watchdog

Unlike most MCUs, NXP's LPC12xx MCUs enable the watchdog timer by default at reset. This option disables this default behavior. For more details, please see the FAQ at http://support.code-red-tech.com/CodeRedWiki/LPC12Watchdog

6.1.7. LPC1102 ISP Pin

Unlike other NXP's MCU, the provision of a pin to trigger entry to NXP's ISP bootloader at reset is not hardwired on the LPC1102. This option allows the generation of default code for providing an ISP pin. For more information, please see NXP's application note, AN11015, "Adding ISP to LPC1102 systems".

6.1.8. Project created

Having selected the appropriate options, you can then click on the Finish button, and the wizard will create your project for you, together with appropriate startup code and a simple main.c file. Build options for the project will be configured appropriately for the MCU that you selected in the project wizard.

You should then be able to build and debug your project, as described in Section 5.2 and Section 5.3.

Chapter 7. Red Suite Memory Editor and User Loadable Flash Driver mechanism

7.1. Introduction

By default, Red Suite provides a standard memory layout for known MCUs. This works well for parts with internal flash and no external memory capability, as it allows linker scripts to be automatically generated for use when building projects, and built in support for programming flash, as well as other debug capabilities.

Red Suite support the editing of the target memory layout used for a project. This allows for the details of external flash to be defined or for the layout of internal RAM to be reconfigured. In addition, it allows a flash driver to be allocated for use with parts with no internal flash, but where an external flash part is connected.

7.2. Memory Editor

The Memory Editor is accessed via the MCU settings dialog, which can be found at:

Project Properties -> C/C++ Build -> MCU settings

This lists the memory details for the selected MCU, and will, by default, display the memory regions that have been defined by Red Suite itself.

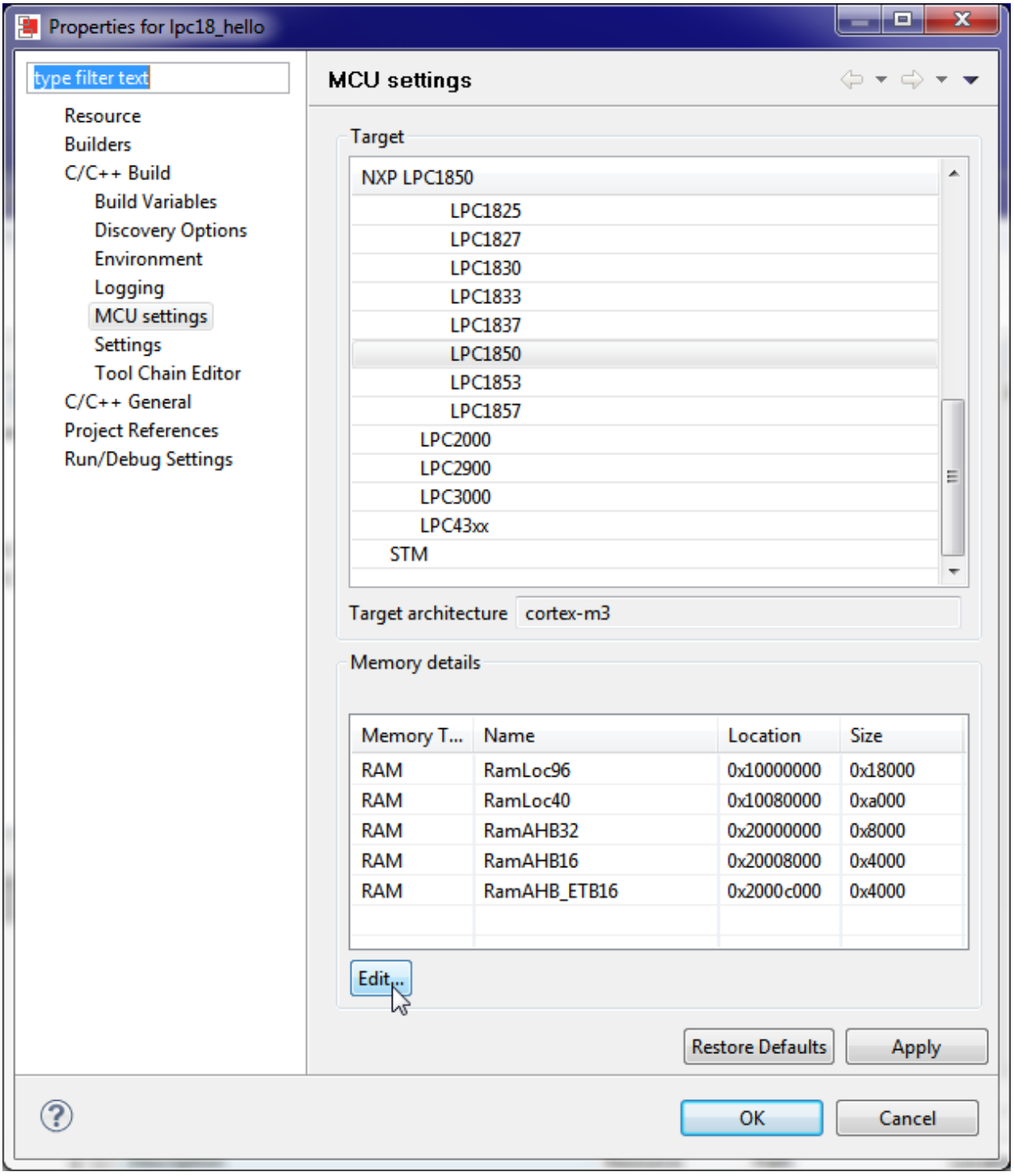


Figure 7.1. Memory editor

7.2.1. Editing a memory configuration

Selecting the **Edit...** button, will launch the **Memory configuration editor** dialog — see Figure 7.2.

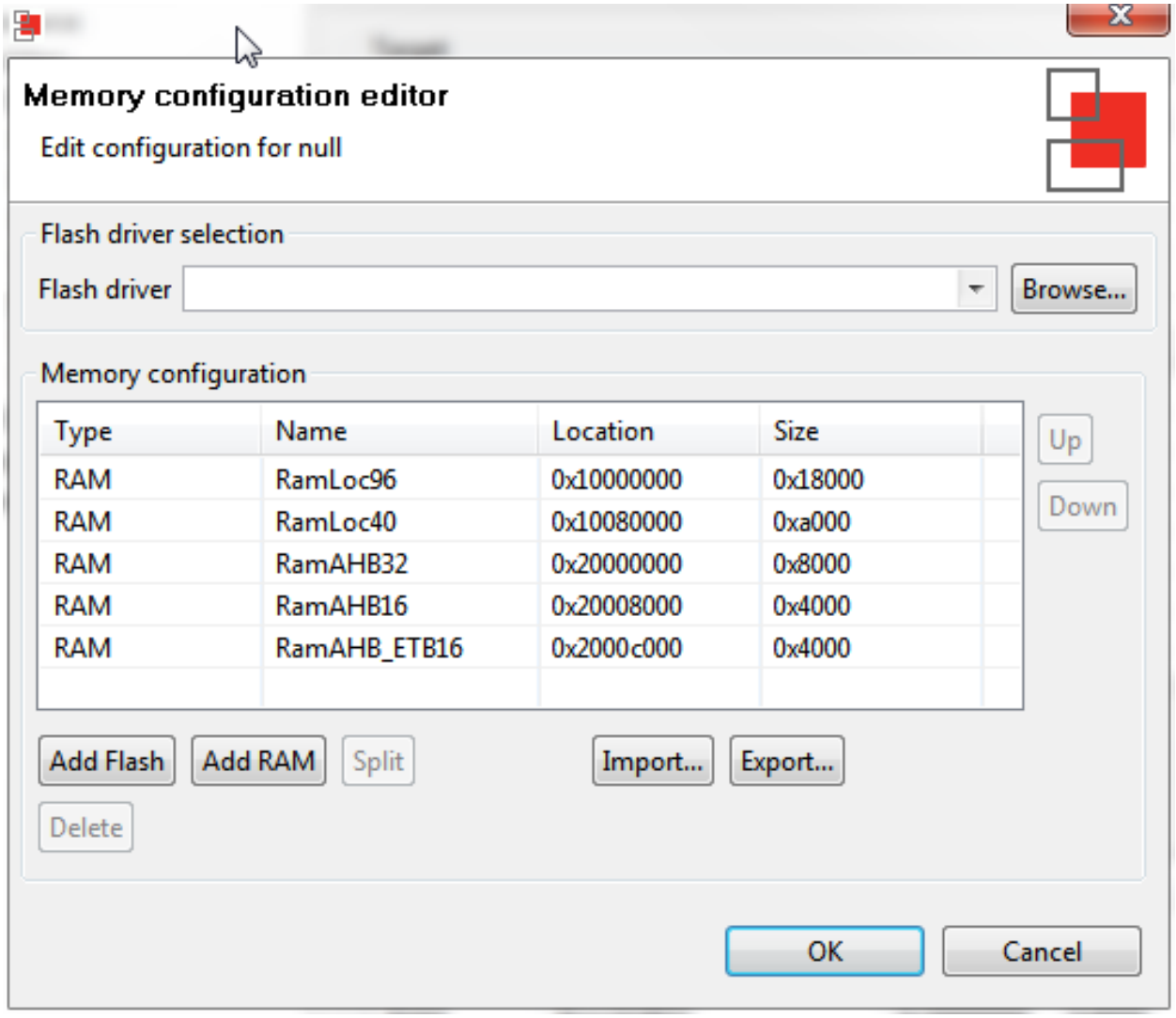


Figure 7.2. Memory configuration editor

Known blocks of memory, their type, base location and size are displayed. Entries can be created/deleted/etc. by using the provided buttons — see Figure 7.2

Table 7.1. Memory editor controls

Button	Details
Add Flash	Add a new memory block of the appropriate type.
Add RAM	Add a new memory block of the appropriate type.
Split	Splits the selected memory block into two equal halves.
Delete	Deletes the selected memory block.
Export	Exports a memory configuration for use in another project.
Import	Imports a memory configuration that has been exported from another project.
Up / Down	Reorder memory blocks. This is important as if there is no flash block, then code will be placed in the first RAM block. And data will be placed in the

Button	Details
	block following the one used for the code (regardless of whether the code block was RAM or Flash).
Browse(Flash driver)	Select the appropriate driver for programming the flash memory specified in the memory configuration. This is only required when the flash memory is external to the MCU. Flash drivers for external flash must have a ".cfx" file extension and must be located in the \bin\flash subdirectory of the Red Suite installation. For more details see User loadable flash drivers (page 33) .

The name, location and size of this new region can be edited in place. Note that when entering the size of the region, you can enter full values in decimal, hex (by prefixing with `0x`), or by specifying the size in kilobytes or megabytes. For example:

- To enter a region size of 32KB, enter `32768`, `0x8000` or `32k`.
- To enter a region size of 1MB, enter `0x100000` or `1m`.

Note that memory regions must be located on 4 byte boundaries, and be a multiple of 4 bytes in size.

The screenshot in Figure 7.3 shows the dialog after the "Add Flash" button has been clicked.

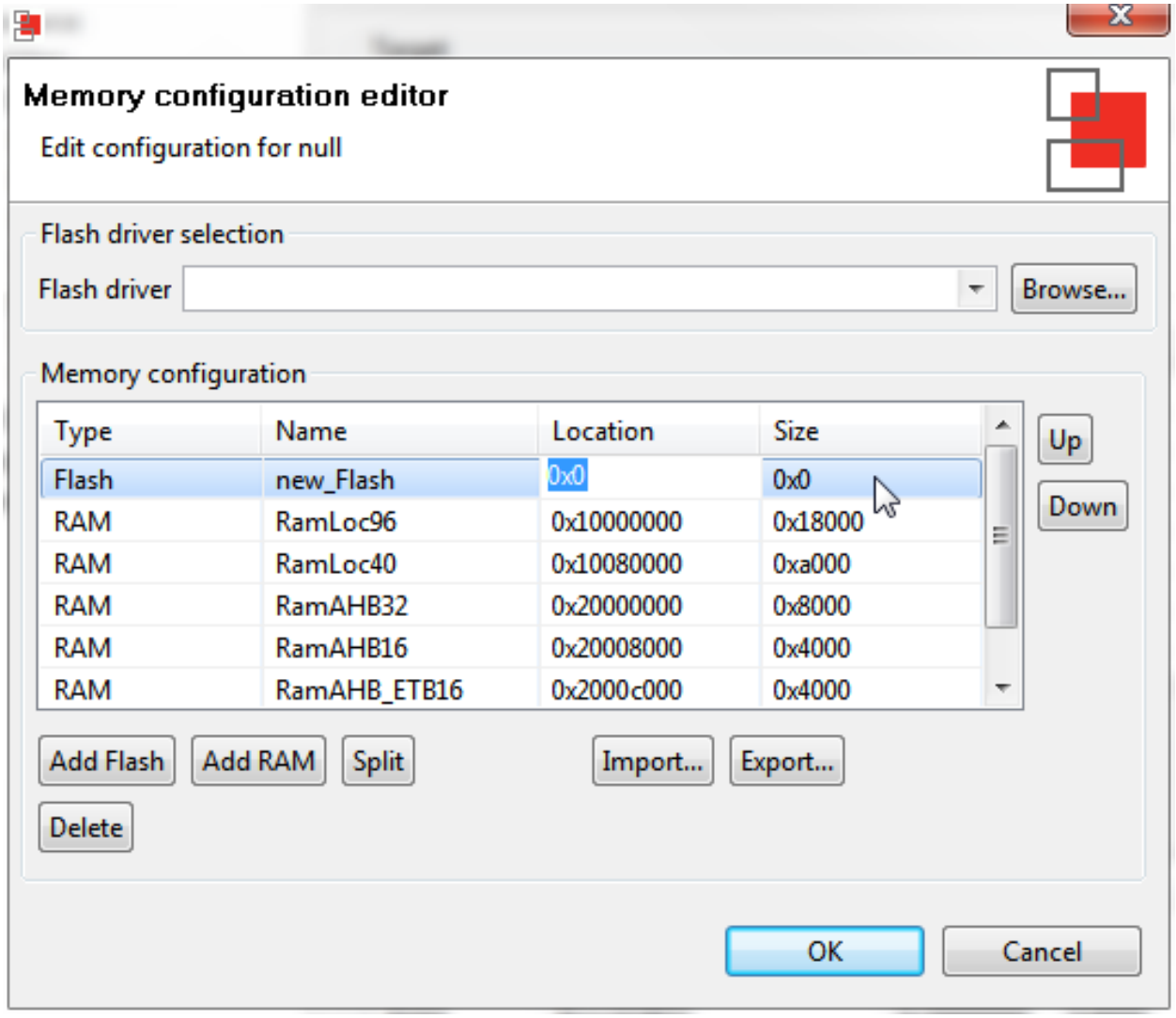


Figure 7.3. Effect of Add Flash

After updating the memory configuration, click **OK** to return to the MCU settings dialog, which will be updated to reflect the new configuration — see Figure 7.4.

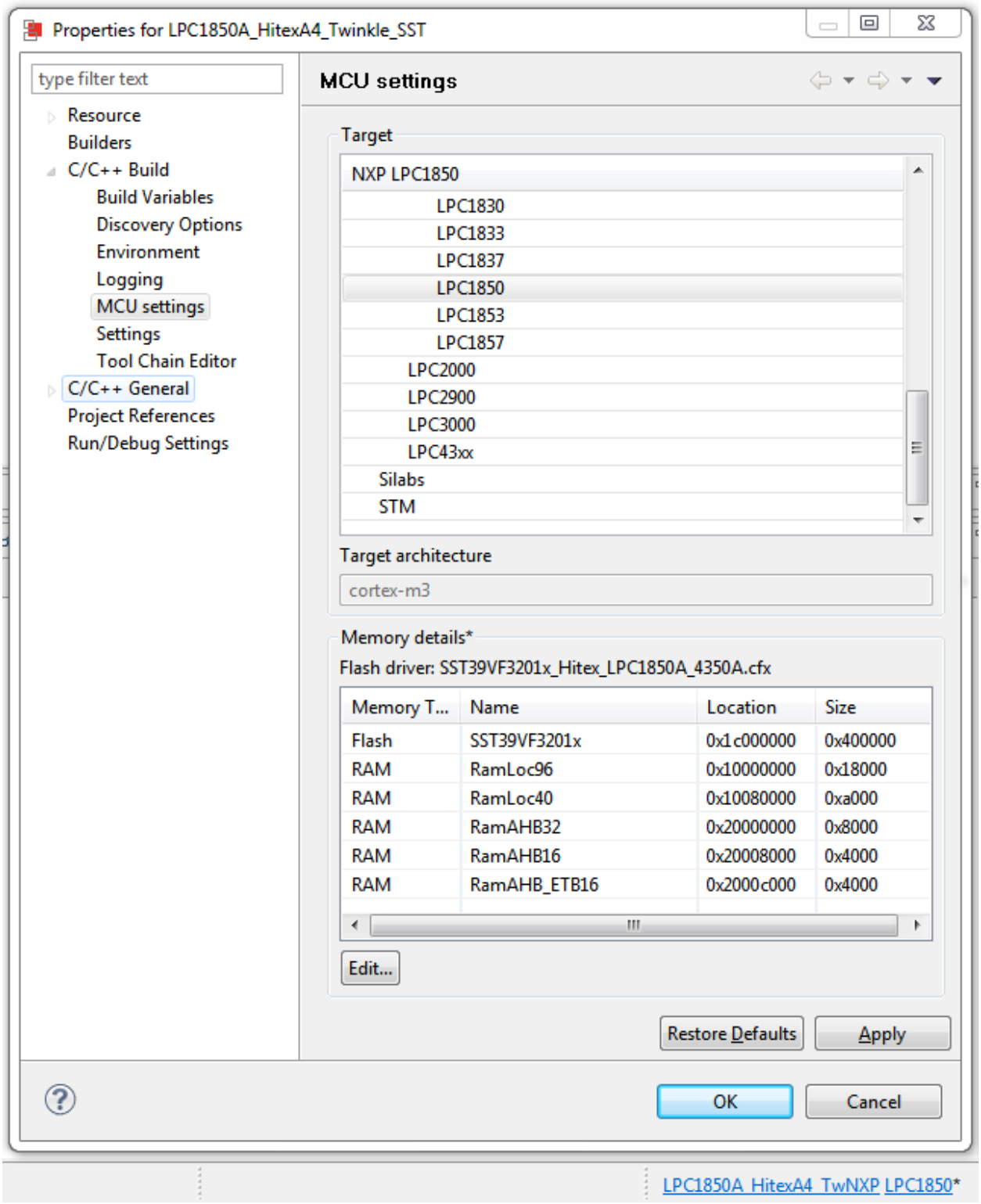


Figure 7.4. Updated MCU Settings

Note that the memory details have been modified, the selected MCU as displayed on the Red Suite ‘Status Bar’ (at the bottom right of the IDE window) will be displayed with an asterisk (*) next to it. This provides an indication that the MCU memory configuration settings for the selected project have been modified, without needing to open up the MCU settings dialog.

7.2.2. Restoring a memory configuration

To restore a memory configuration, simply reselect the MCU type in the MCU Settings properties page.

7.2.3. Copying memory configurations

Memory configurations can be exported for import into another project. Use the Export and Import buttons for this purpose.

7.3. User loadable flash drivers

Flash drivers for external flash must have a `.cfx` file extension and must be located in the `/bin/flash` subdirectory of the Red Suite installation.

For example:

- The `SST39VF3201x_Hitex_LPC1850A_4350A.cfx` driver is for use programming the external 4MB parallel flash fitted to the Hitex LP1850A/LPC4350A evaluation board (either a SST39VF3201 or SST39VF3201B device). This memory has a base address of `0x1c000000`.

Note that the `SST39VF3201B_Hitex_LPC1850A_4350A.cfx` driver is also provided with Red Suite 4.2.1 for compatibility with memory configurations created with Red Suite 4.2.0. This is only compatible with the SST39VF3201B flash device and is not recommended for use with new projects. Use the `SST39VF3201x_Hitex_LPC1850A_4350A.cfx` driver instead.

- The `LPC18_43_SPIFI_8MB_64KB.cfx` driver is for use programming external serial flash fitted to the Hitex LP1850A/LPC4350A evaluation board. This memory is located at a base address of `0x14000000` is 8MB in size, and has 64KB sectors.

Source code for external flash drivers is also provided in the `/Examples/FlashDrivers` subdirectory of the Red Suite installation.

Note that the `/bin/flash` subdirectory may also contain some drivers for the built in flash on some MCUs. It should be clear from the filenames as to which these are. Do not try to use these drivers for external flash on other MCUs!

For more details of the user loadable flash driver mechanism, please see the FAQ at http://support.code-red-tech.com/CodeRedWiki/UserLoadableFlashDrivers

7.4. Importing memory configurations via New Project Wizards

The New Project Wizards for LPC18xx and LPC43xx parts allows a memory configuration file (as exported from the Memory Configuration Editor, as detailed in Editing a memory configuration (page 28)) to be imported at the time of creating a project.

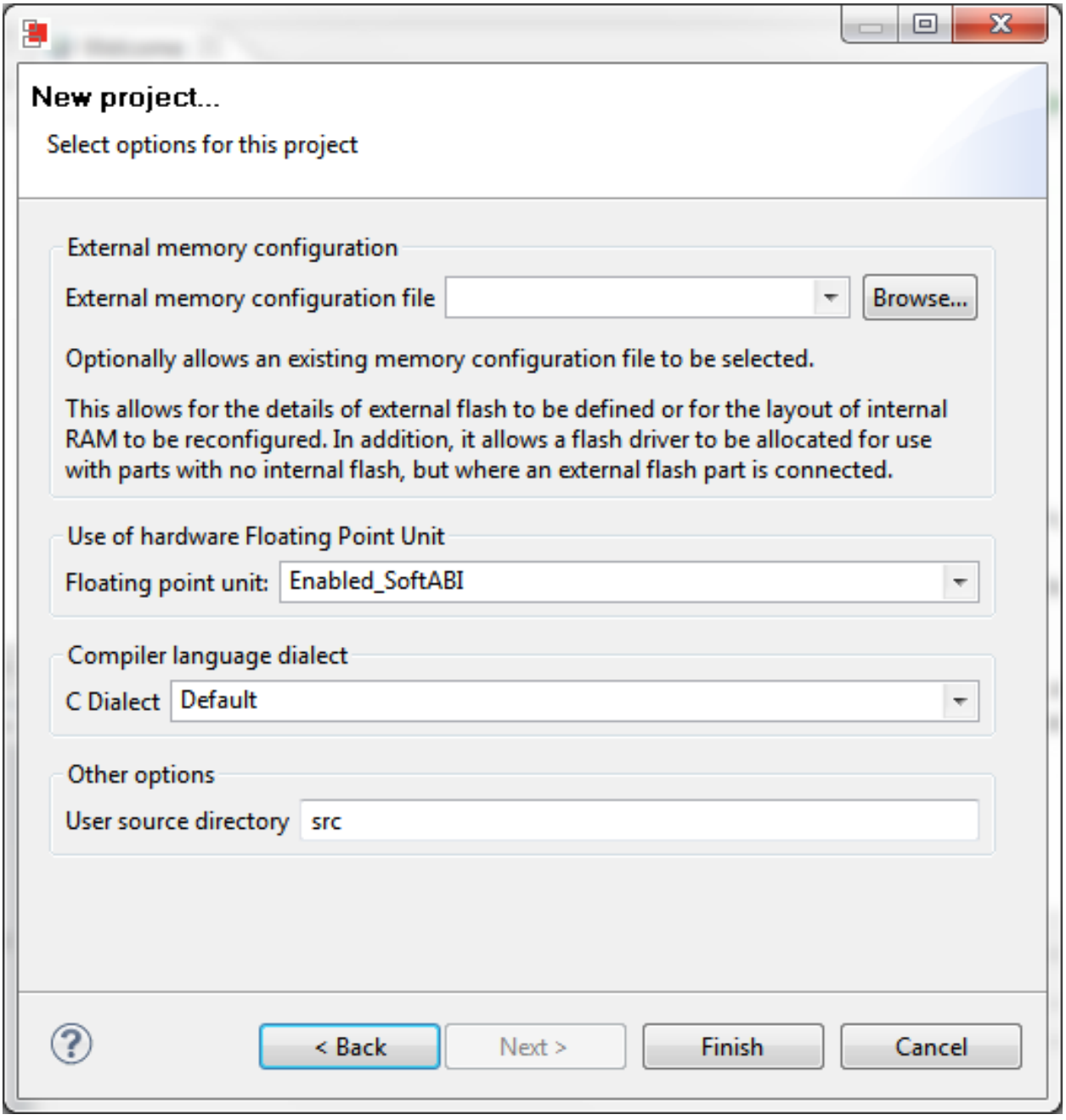


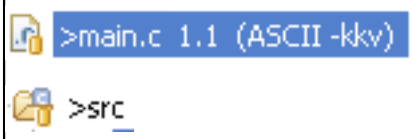
Figure 7.5. New project wizard

A default set of memory configurations for importing can be found in the directory

```
<install_dir>\redsuite\Wizards\MemConfigs\NXP.
```

Chapter 8. Appendix A File Icons

Table 8.1. File Icons

Icon	Meaning
	C language source file.
	C language header file.
	Source folder (generally use for source files, including headers)
	Folder (use for none-source files)
	Source folder with modified build properties
	Source file with modified build properties
	Project Makefile. Note that Red Suite may automatically generate these.
	C language source file that has been excluded from the project build. Hollowed out character in icon.
	General text file. Often used for files such as linker script files which end with the extension ‘.ld’.
	The blue double arrows at the top of the icon denote that this particular source file has different project build properties than the rest of the project.
	The orange/gold ‘storage drive’ denotes that this file is connected to a CVS repository.
	The right arrow denotes that the file or directory has been modified locally and may need committing to the CVS repository. I.e. The local copy is more up to date than the repository.
	This is an executable file.
	Directory containing executables.
	C Project Directory (Project Explorer View)
	C Project Directory linked to a CVS repository. (Project Explorer View)

Chapter 9. Appendix B Glossary of Terms

Table 9.1. Explanation of the meaning of terms used in this document.

Term	Meaning
Red Suite™	The Code Red Technologies IDE (Integrated Development Environment) based on Eclipse with our own extensions for embedded development.
LPCXpresso IDE	Reduced functionality version of Red Suite for specific NXP microcontrollers.
Red Suite NXP Edition	Version of Red Suite for NXP microcontrollers only, with variants with 256KB and 512KB code download limits.
Red Probe™	The Code Red Technologies debug probe that is compatible with both SWD/SWV and JTAG debug targets. For Cortex-Mx based devices, it will normally default to using SWD/SWV. For other ARM processors it will default to using JTAG.
Red Probe+™	Higher performance version of Red Probe.
LPC-Link	Low end debug probe for NXP microcontrollers, provided as part of NXP LPCXpresso evaluation boards
Redlib™	The Code Red Technologies C90 runtime library (non-GNU). Generally provides reduced code size compared to Newlib.
Newlib	Open source C99 runtime library. Can optionally be used instead of Redlib for C projects, and required for C++ projects.
SWD	Serial Wire Debugging (Single Wire Debugging).This is a debug connection technology available on the Cortex-M3 that allows debug through just 2-wires unlike 6 for JTAG.
SWV (Red Trace™)	Serial Wire Viewing. This is an additional feature to SWD that allows Red Suite to give real-time tracing visualization and views from Cortex-M3 based devices. SWV is only available if SWD is being used.
ELF	Executable and Linking Format. This is the object code file format used by our development toolchain and most microprocessor toolchains.
DWARF	Debug with Attributed Record Format. Developed along with ELF but actually independent of the Object file format. DWARF is a format defined for carrying debug information in object files.
Workspace	Red Suite organizes groups of projects into a ‘Workspace’. A workspace is stored as a directory on your host PC and has subdirectories containing individual projects.
Project	A collection of source files and settings

Term	Meaning
Perspective	In Red Suite, a perspective is a particular collection of ‘Views’ that are grouped together to be suitable for a particular use. For example the ‘C/C++ programming’ perspective and the ‘Debug’ perspective.
View	A ‘View’ is a window in Red Suite that shows a particular file or activity. A ‘Perspective’ is the layout of many ‘Views’.
Semihosting	The ability to use IO on your debugger host system for your target embedded system. For example a ‘printf’ will appear in the console window of the debugger.
Debug Target	The development board (evaluation board) or debug probe connected to a board.
Build artifact	The ‘Build Artifact’ is the final product of all the build steps (with the exception of any post dump files produced with ‘objdump’).. The Build Artifact is correctly set as the name of a new project by default, but may be changed manually. This can be useful, if for example you have copied an existing project (cloned it) and you want the new project executable to have the name of your new copy of the project.