User's Manual

e^2 studio v7.0

Integrated Development Environment

User's Manual: Getting Started Guide

Target Device
RX, RL78, RH850 and RZ Family

Renesas Electronics
www.renesas.com

Rev.1.00 July 2018

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.

   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.

   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(2012.4)

How to Use This Manual

This manual describes the role of the e^2 studio integrated development environment for developing applications and systems and provides an outline of its features.

e^2 studio is an integrated development environment (IDE) for RX family, RL78 family and RZ family integrating the necessary tools for the development phase of software (e.g. design, implementation, and debugging) into a single platform.

By providing an integrated environment, it is possible to perform all development using just this product, without the need to use many different tools separately.

Readers This manual is intended for users who wish to understand the functions of the e^2 studio and design software and hardware application systems.

Purpose This manual aims to provide user with the explanation of the functions provided in e^2 studio when they commence the development of their hardware and software systems using the targeted devices.

Organization This manual can be broadly divided into the following units.

CHAPTER 1 GENERAL
CHAPTER 2 INSTALLATION
CHAPTER 3 PROJECT GENERATION
CHAPTER 4 BUILD
CHAPTER 5 DEBUG
CHAPTER 6 HELP

How to Read This Manual It is assumed that the readers of this manual have general knowledge of electricity, logic circuits, and microcontrollers.

Conventions

Data significance:	Higher digits on the left and lower digits on the right
Active low representation:	XXX (overscore over pin or signal name)
Note:	Footnote for item marked with Note in the text
Caution:	Information requiring particular attention
Remark:	Supplementary information
Numeric representation:	Decimal ... XXXX Hexadecimal ... 0xXXXX

TABLE OF CONTENTS

CHAPTER 1. GENERAL

Renesas eclipse embedded studio (known as “e^2 studio”) is a complete, state of the art development environment supporting Renesas embedded micro-controllers. It is developed based on a popular open-source Eclipse CDT (C/C++ Development Tooling) project that covers build (e.g. editor, compiler and linker control) and debug phase with an extended GDB interface support.

This chapter describes the system configuration and operating environment for e^2 studio IDE to develop applications for the RX family series microcontrollers as example.

1.1. System Configuration

Below is an example of a typical system configuration.

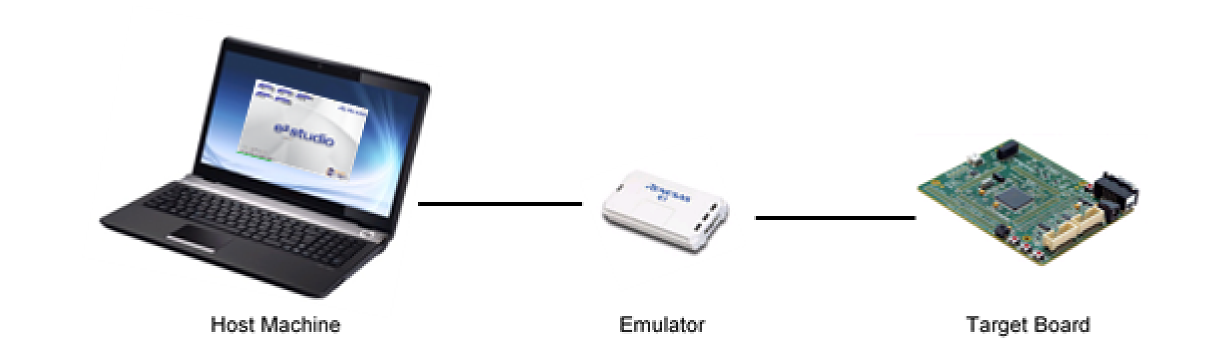


Figure 1-1 System Configuration

1.2. Operating Environment

Below are the system requirements for this product.

1.2.1. System Requirements

Hardware Environment:

Processor:	At least 1GHz (support hyper-threading/multi-core CPU)
Main Memory:	At least 1GB (2GB or larger is recommended, especially for Windows 64-bit OS)
Display:	Resolution at least 1,024 x 768; at least 65,536 colors
Interface:	USB 2.0 (High-speed/Full-speed). High-speed is recommended.

Software Environment:
Windows 7 (32/64-bit OS), Windows 8.1 (32/64-bit OS) and Windows 10 (32/64-bit OS)

1.2.2. Supported Toolchain

1.2.2.1. Supported Compiler

Renesas C/C++ compiler package for RX family

Renesas C compiler package for RL78 family

Gnu ARM Embedded Toolchain for RZ and Renesas Synergy

GNURX Windows Toolchain (ELF)

GNURL78 Windows Toolchain (ELF)

1.2.2.2. Supported Emulator

E2 emulator Lite (RX, RL78), E1 (RX, RL78, RH850), E2 (RX, RL78, RH850), E20 (RX) and Segger J-Link (RX, RZ)

1.2.2.3. Supported Simulator

Renesas Simulator (RX), GDB Simulator (RL78)

CHAPTER 2. INSTALLATION

The latest e^2 studio IDE installer package can be downloaded from Renesas website for free. User has to login to the Renesas account (in MyRenesas page) for the software download.

This chapter describes the installation and un-installation for the e^2 studio IDE.

e^2 studio installer can be used to upgrade e^2 studio as well as new installation. ‘Modify’ function is available if you chose an existing installation folder at step (2).
However, it does not support update between major versions such as from V5.4 to V6.0, or from V6.3 to V7.0. Please uninstall the earlier versions before installation. Alternatively, install new e^2 studio into a new folder if you would like to keep earlier versions.

2.1. Installation of e^2 studio IDE

(1) Double-click on e^2 studio installer to invoke the e^2 studio installation wizard page.
Click the [Next] button to continue.

(2) Install Folder

The default installation location is set to: “C:\Renesas\e2_studio”. Input install folder directly to textbox or click [Browse…] button to modify it.

Click the [Next] button to continue.

Note1: If you would like to have multiple versions of e^2 studio, please specify new folder here.
Note2: Multi-byte characters cannot be used for e^2 studio installation folder name.

(3) Device Families

Select Devices Families to install. Click the [Next] button to continue.

(4) Extra Components

Select Extra Components (i.e. Language packs, SVN & Git support, RTOS support…) to install. Click the [Next] button to continue.

(5) Components

Select Components and click the [Next] button to continue.

(6) Additional Software

Select additional software (i.e. compilers, utilities) and click the [Next] button to continue.

Note: With no Internet access available, additional software installation can be skipped because software catalog cannot be downloaded. You can still continue installation, anyway.

(7) License Agreement

Read and accept the software license agreement to proceed with the [Next] button.

Please note that user has to accept the license agreement, otherwise installation cannot be continued.

(8) Shortcuts

Select shortcut name for start menu and click [Next] button to continue.

Note: If you already have installed e^2 studio in another location, it is recommended to rename to distinguish from the other e^2 studio(s).

(9) Summary

Click the [Install] button to install the Renesas e^2 studio IDE.

(10) Installing…

The installation is performed. Based on selected items of Addition Software, new dialogs are opened to proceed with installation for these software.

(11) Results

Click [OK] button to complete the installation.

2.2. Un-installation of e^2 studio IDE

User can uninstall e^2 studio program following the typical steps to uninstall a program in Window OS.

(1) Search for [Apps & features] in Window Search Box. Click on the search result to go to [Apps & features].

(2) From the currently installed programs list, choose "e^2 studio" and click the [Uninstall] button.

(3) Click the [Uninstall] button again to confirm the deletion of e^2 studio.

At the end of the un-installation, e^2 studio IDE will be deleted from the installed location and Windows short-cuts menu are removed.

Note:

If you have installed e^2 studio at multiple locations, you may not able to find uninstaller in "Apps & features" of Control Panel. In such cases, launch e^2 studio uninstaller located at :

{e^2 studio installed folder}/uninstall/uninstall.exe

2.3. Upgrade versions using e^2 studio installer

[Check for Updates] (mentioned in chapter 2.4) does not install new features. Therefore, you need e^2 studio installer to perform upgrade e^2 studio version. However, upgrading over major versions (i.e. the number increase in the major digit, for example, version up from v.4.3.0.008 to v.5.2.0.020) is not recommended because plugins compatibility and integrity may not guaranteed between different eclipse platform versions. Prior to the IDE upgrade, user must uninstall the old version IDE. To keep both old and new IDE versions, user can create new folder as installation destination for the new version IDE.

e^2 studio installer supports minor version update (i.e. number increase in the minor digit, for example, version up from v4.0.0.21 to v4.0.0.23). Please choose “Modify” at

2.4. Update plugins over Internet

Following procedure tells how to update existing plugins, **not to install new features**. Please use e2 studio installer to upgrade newer versions.

2.4.1. Online Minor Version Update

This section illustrates an example on the steps to launch the online minor version update

(1) From the [Help] menu, click the [Check for Updates] to display the [Available update] panel.

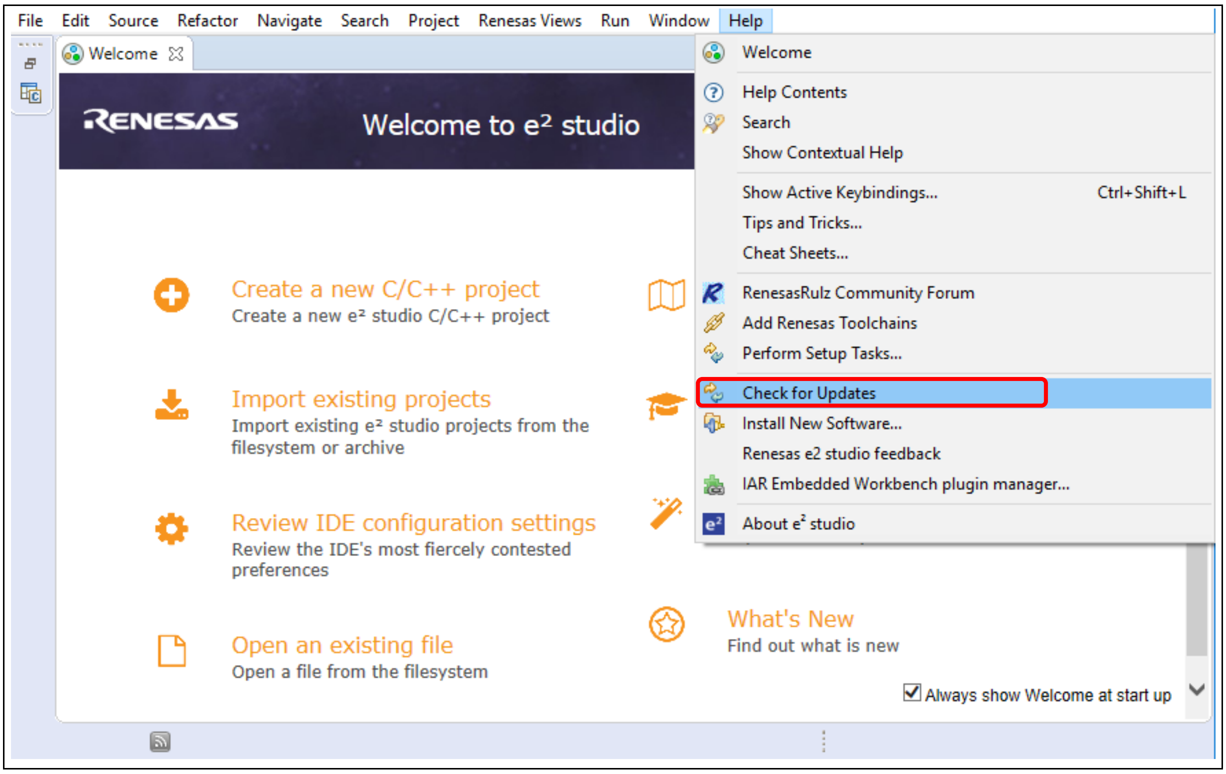
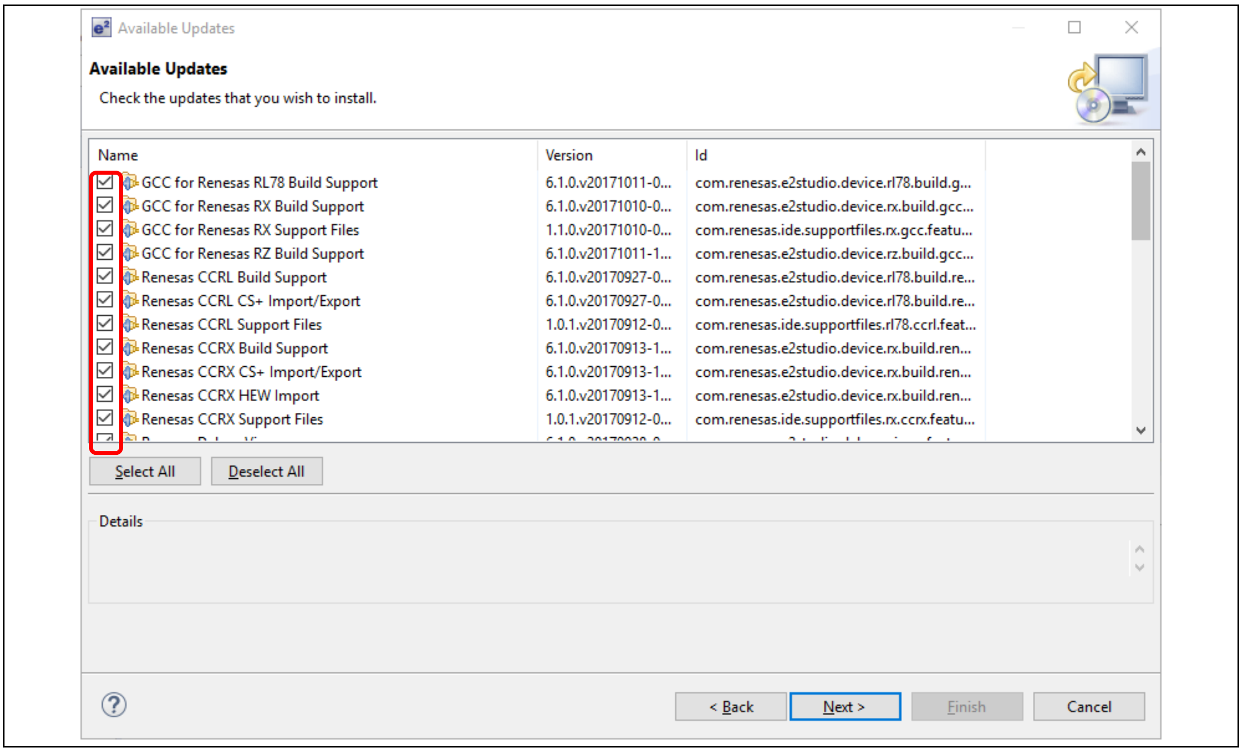

File
Edit
Source
Refactor
Navigate
Search
Project
Renesas Views
Run
Window
Help
Welcome
RENESAS
Welcome to e² studio
Create a new C/C++ project
Create a new e² studio C/C++ project
Import existing projects
Import existing e² studio projects from the filesystem or archive
Review IDE configuration settings
Review the IDE's most fiercely contested preferences
Open an existing file
Open a file from the filesystem
What's New
Find out what is new
☑ Always show Welcome at start up
Welcome
Help Contents
Search
Show Contextual Help
Show Active Keybindings... Ctrl+Shift+L
Tips and Tricks...
Cheat Sheets...
RenesasRulz Community Forum
Add Renesas Toolchains
Perform Setup Tasks...
Check for Updates
Install New Software...
Renesas e2 studio feedback
IAR Embedded Workbench plugin manager...
About e² studio

Figure 2-1 [Check for Updates] Menu

(2) By default, all the software components are selected in the [Available Updates] panel. This allows user to update them all to the latest version. (An example is shown in Figure 2-2). Click the [Next] button to proceed.



Available Updates

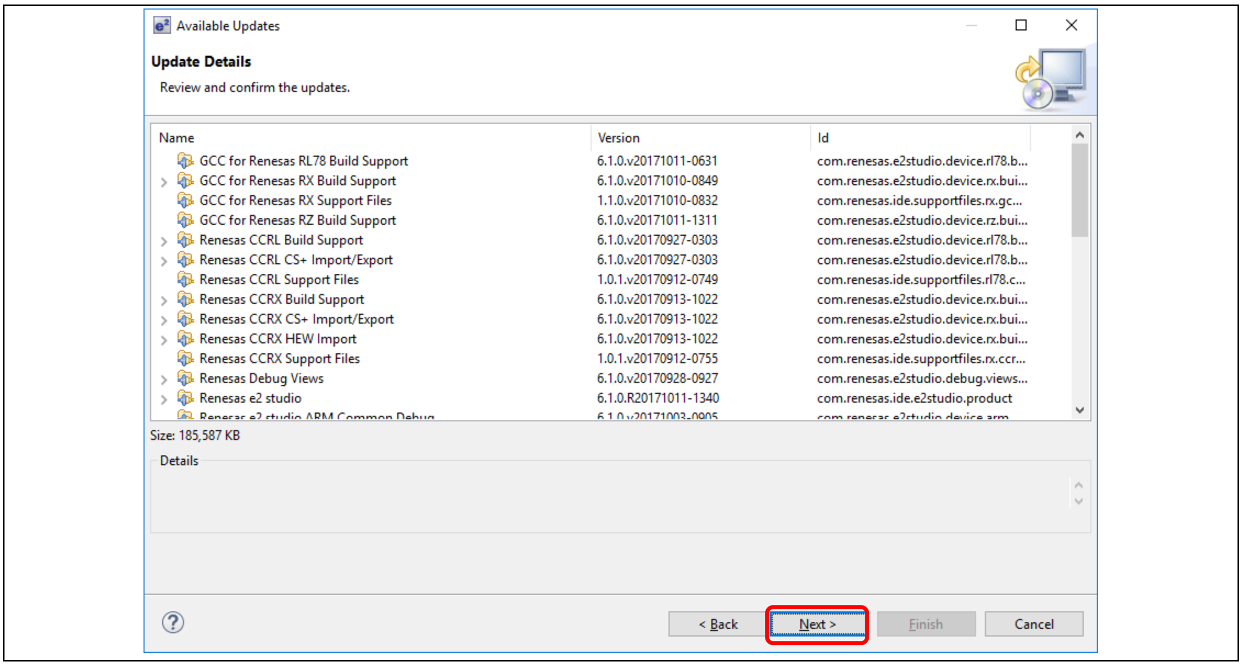
Available Updates

Check the updates that you wish to install.

Name	Version	Id
☑ GCC for Renesas RL78 Build Support	6.1.0.v20171011-0...	com.renesas.e2studio.device.rl78.build.g...
☑ GCC for Renesas RX Build Support	6.1.0.v20171010-0...	com.renesas.e2studio.device.rx.build.gcc...
☑ GCC for Renesas RX Support Files	1.1.0.v20171010-0...	com.renesas.ide.supportfiles.rx.gcc.featu...
☑ GCC for Renesas RZ Build Support	6.1.0.v20171011-1...	com.renesas.e2studio.device.rz.build.gcc...
☑ Renesas CCRL Build Support	6.1.0.v20170927-0...	com.renesas.e2studio.device.rl78.build.re...
☑ Renesas CCRL CS+ Import/Export	6.1.0.v20170927-0...	com.renesas.e2studio.device.rl78.build.re...
☑ Renesas CCRL Support Files	1.0.1.v20170912-0...	com.renesas.ide.supportfiles.rl78.ccrl.feat...
☑ Renesas CCRX Build Support	6.1.0.v20170913-1...	com.renesas.e2studio.device.rx.build.ren...
☑ Renesas CCRX CS+ Import/Export	6.1.0.v20170913-1...	com.renesas.e2studio.device.rx.build.ren...
☑ Renesas CCRX HEW Import	6.1.0.v20170913-1...	com.renesas.e2studio.device.rx.build.ren...
☑ Renesas CCRX Support Files	1.0.1.v20170912-0...	com.renesas.ide.supportfiles.rx.ccrx.featu...

Figure 2-2 e^2 studio – Available Updates panel (1/3)

(3) Select the [Next] button to continue the update.



Available Updates

Update Details

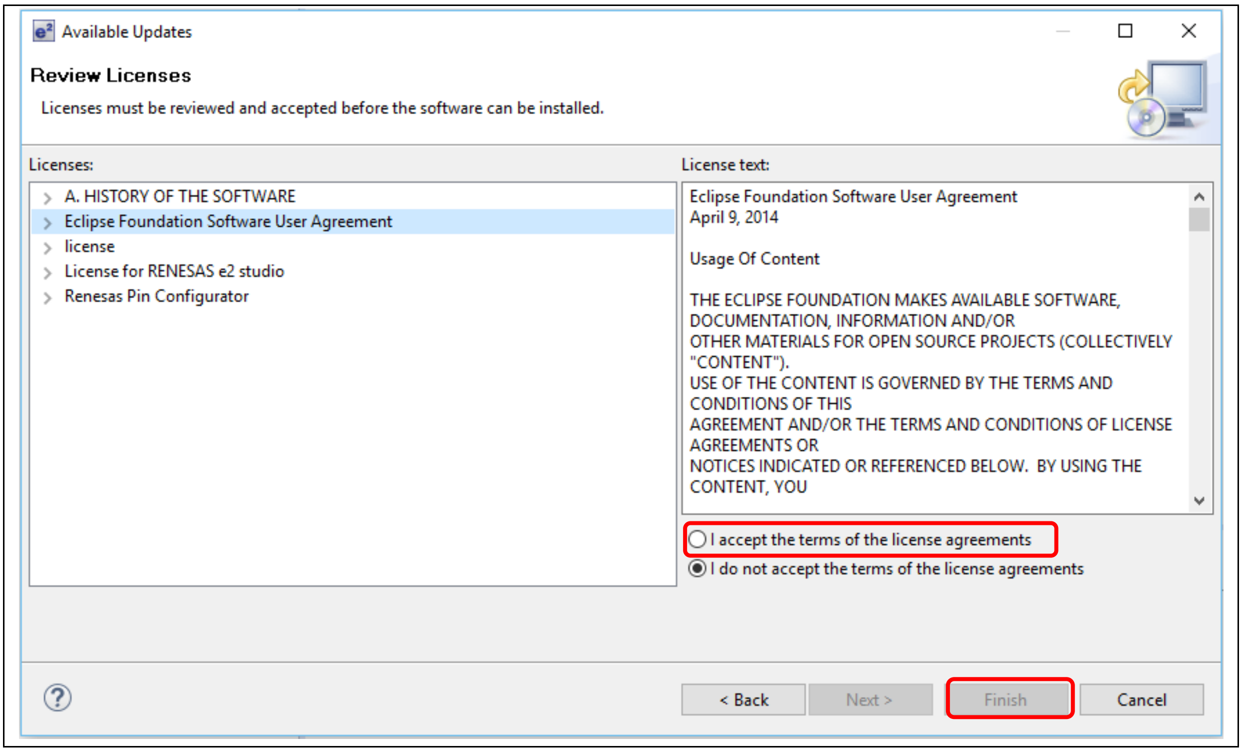
Review and confirm the updates.

Name	Version	Id
GCC for Renesas RL78 Build Support	6.1.0.v20171011-0631	com.renesas.e2studio.device.rl78.b...
GCC for Renesas RX Build Support	6.1.0.v20171010-0849	com.renesas.e2studio.device.rx.bui...
GCC for Renesas RX Support Files	1.1.0.v20171010-0832	com.renesas.ide.supportfiles.rx.gc...
GCC for Renesas RZ Build Support	6.1.0.v20171011-1311	com.renesas.e2studio.device.rz.bui...
Renesas CCRL Build Support	6.1.0.v20170927-0303	com.renesas.e2studio.device.rl78.b...
Renesas CCRL CS+ Import/Export	6.1.0.v20170927-0303	com.renesas.e2studio.device.rl78.b...
Renesas CCRL Support Files	1.0.1.v20170912-0749	com.renesas.ide.supportfiles.rl78.c...
Renesas CCRX Build Support	6.1.0.v20170913-1022	com.renesas.e2studio.device.rx.bui...
Renesas CCRX CS+ Import/Export	6.1.0.v20170913-1022	com.renesas.e2studio.device.rx.bui...
Renesas CCRX HEW Import	6.1.0.v20170913-1022	com.renesas.e2studio.device.rx.bui...
Renesas CCRX Support Files	1.0.1.v20170912-0755	com.renesas.ide.supportfiles.rx.ccr...
Renesas Debug Views	6.1.0.v20170928-0927	com.renesas.e2studio.debug.views...
Renesas e2 studio	6.1.0.R20171011-1340	com.renesas.ide.e2studio.product
Renesas e2 studio ARM Common Debug	6.1.0.v20171003-0905	com.renesas.e2studio.device.arm

Size: 185,587 KB

Figure 2-3 e^2 studio – Available Updates panel (2/3)

(4) Read and check the software license agreement. Click the [Finish] button to complete update.



Available Updates

Review Licenses

Licenses must be reviewed and accepted before the software can be installed.

Licenses:

- A. HISTORY OF THE SOFTWARE
- Eclipse Foundation Software User Agreement
- license
- License for RENESAS e2 studio
- Renesas Pin Configurator

License text:

Eclipse Foundation Software User Agreement
April 9, 2014

Usage Of Content

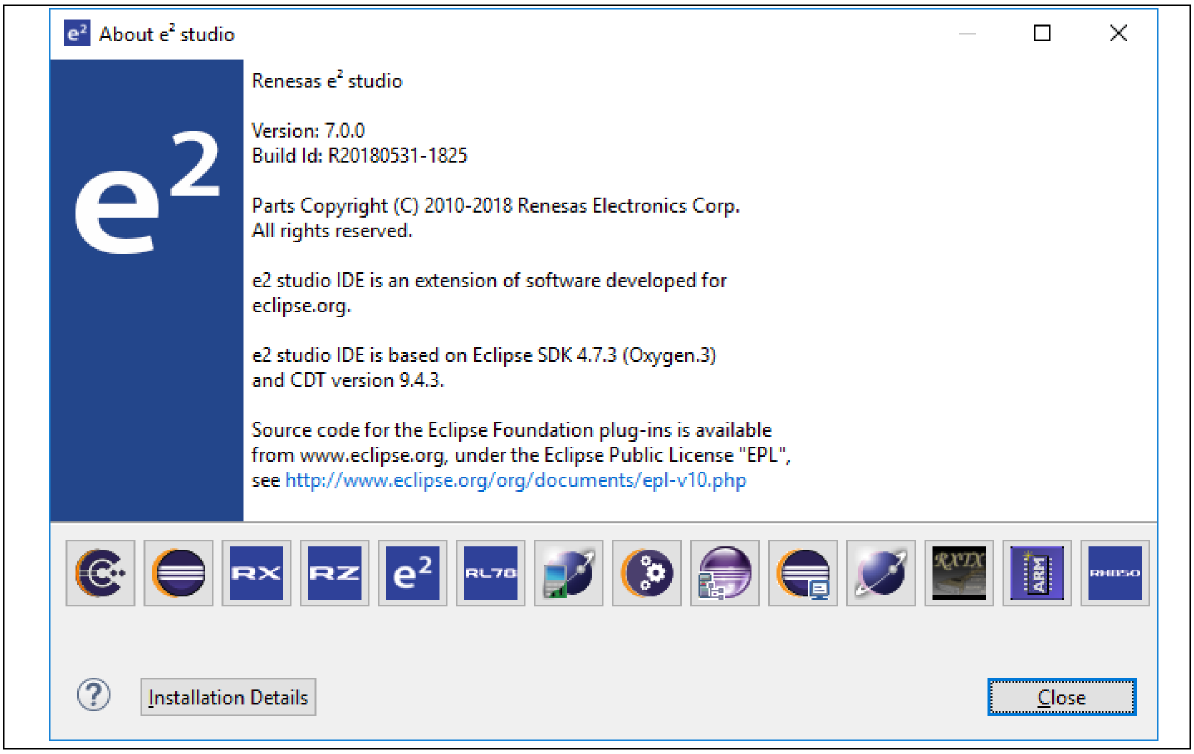
THE ECLIPSE FOUNDATION MAKES AVAILABLE SOFTWARE, DOCUMENTATION, INFORMATION AND/OR OTHER MATERIALS FOR OPEN SOURCE PROJECTS (COLLECTIVELY "CONTENT"). USE OF THE CONTENT IS GOVERNED BY THE TERMS AND CONDITIONS OF THIS AGREEMENT AND/OR THE TERMS AND CONDITIONS OF LICENSE AGREEMENTS OR NOTICES INDICATED OR REFERENCED BELOW. BY USING THE CONTENT, YOU

○ I accept the terms of the license agreements

● I do not accept the terms of the license agreements

Figure 2-4 e^2 studio – Available Updates panel (3/3)

(5) Click the [Help] → [About e^2 studio] to confirm the updated version.



About e^2 studio

Renesas e^2 studio

Version: 7.0.0
Build Id: R20180531-1825

Parts Copyright (C) 2010-2018 Renesas Electronics Corp.
All rights reserved.

e2 studio IDE is an extension of software developed for eclipse.org.

e2 studio IDE is based on Eclipse SDK 4.7.3 (Oxygen.3) and CDT version 9.4.3.

Source code for the Eclipse Foundation plug-ins is available from www.eclipse.org, under the Eclipse Public License "EPL", see http://www.eclipse.org/org/documents/epl-v10.php

Figure 2-5 e^2 studio – About e^2 studio panel

2.4.2. Offline Minor Version Update

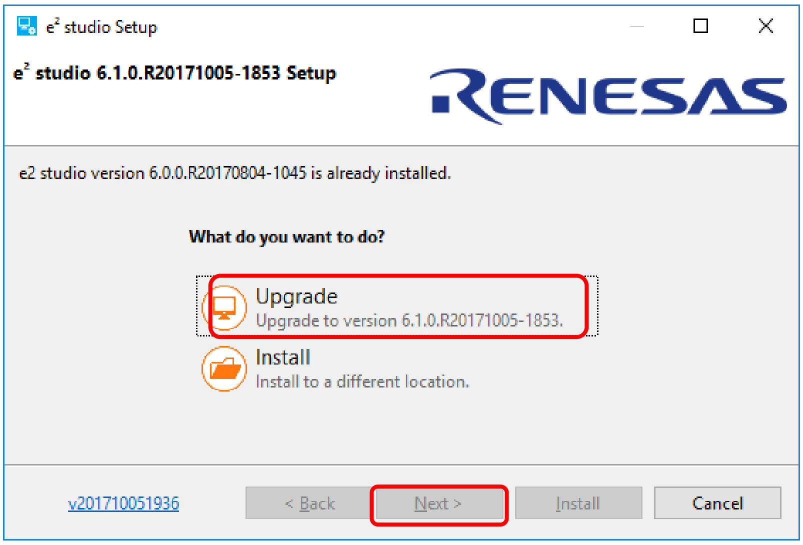
This section illustrates how to update e² studio with the upgrade function of the installer.

(1) Download the desired new version of e² studio offline installer from the following Renesas URL: http://www.renesas.com/e2studio_download

Note: Offline version update using ‘Differential Update program’ is available with e^2 studio Ver3.x or older versions.

(2) Double-click to run the installer file downloaded in step (1). The installer will detect existing version and user can choose to upgrade or install new e^2 studio version to a different folder.

Click [Upgrade], [Next] to begin the upgrading.



e^2 studio Setup

e^2 studio 6.1.0.R20171005-1853 Setup

e2 studio version 6.0.0.R20170804-1045 is already installed.

What do you want to do?

Upgrade
Upgrade to version 6.1.0.R20171005-1853.

Install
Install to a different location.

Figure 2-6 Upgrade e^2 studio from offline installer

(3) Follow the steps shown in Section 2.1 Installation of e^2 studio IDE. Step (2) Install Folder is skipped since Upgrade option will use the same destination folder as existing e^2 studio.

2.5. Installation of Compiler Package

V5.0 or newer e^2 studio installer is capable of installing compiler packages automatically during e^2 studio installation with valid Internet connection. However, in situation where Internet connection is not available during e^2 studio installation, compiler packages can be installed later from compiler package installation files from the web site shown below. This step is similar with e^2 studio V4.x or older.

Renesas Compiler Package download sites:

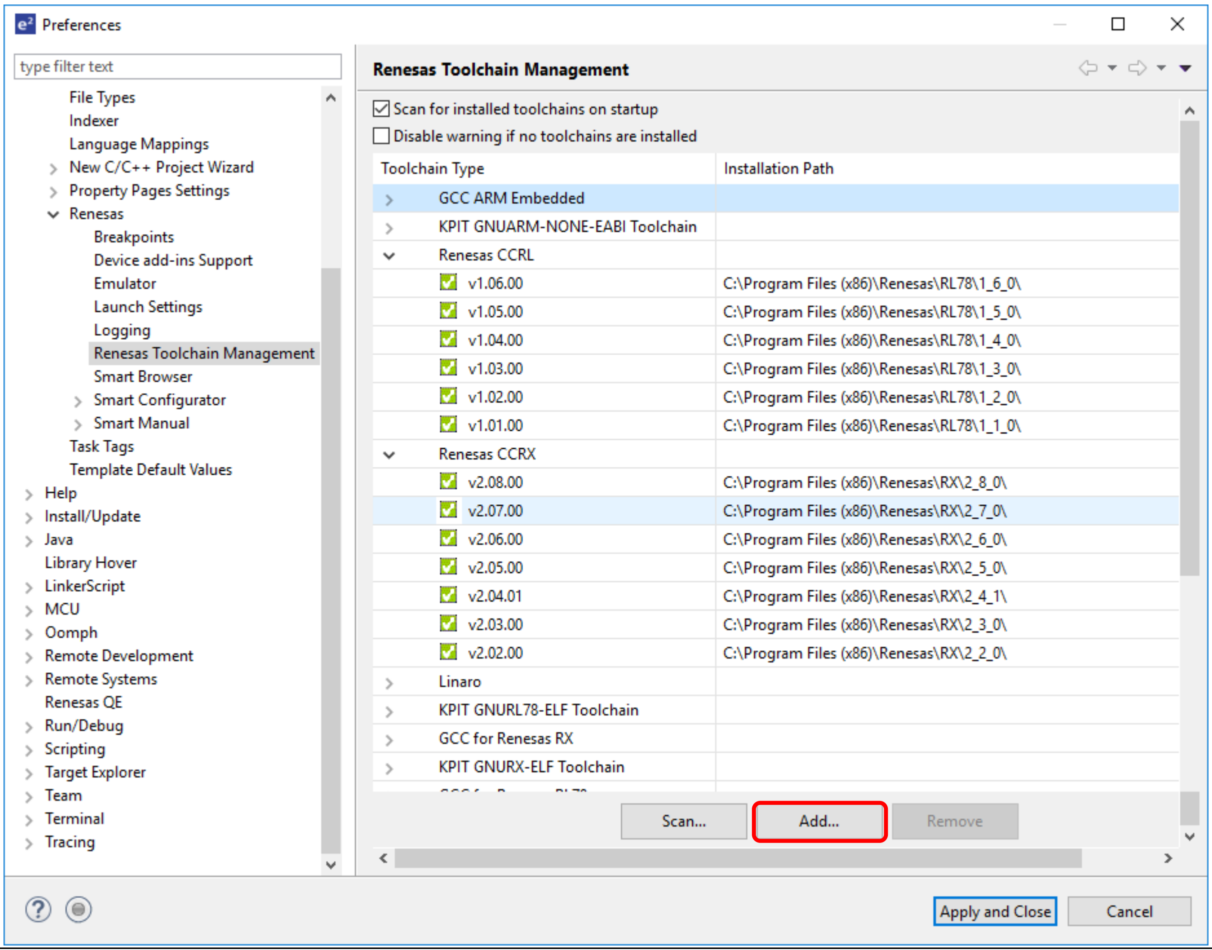
For RX Family: http://www.renesas.com/rx_c

For RL78 Family: http://www.renesas.com/rl78_c

GNU Toolchain download site:
https://gcc-renesas.com/

To check for compilers already installed, click from the toolbar or click [Help] → [Add Renesas Toolchains] to open Renesas Toolchain Management as shown below. Check the desired toolchain to integrate it in e^2 studio.

If desired compiler is not listed, Click [Add…] and specify the installed location.



Renesas Toolchain Management

☑ Scan for installed toolchains on startup
☐ Disable warning if no toolchains are installed

Toolchain Type	Installation Path
GCC ARM Embedded	
KPIT GNUARM-NONE-EABI Toolchain	
Renesas CCRL	
☑ v1.06.00	C:\Program Files (x86)\Renesas\RL78\1_6_0\
☑ v1.05.00	C:\Program Files (x86)\Renesas\RL78\1_5_0\
☑ v1.04.00	C:\Program Files (x86)\Renesas\RL78\1_4_0\
☑ v1.03.00	C:\Program Files (x86)\Renesas\RL78\1_3_0\
☑ v1.02.00	C:\Program Files (x86)\Renesas\RL78\1_2_0\
☑ v1.01.00	C:\Program Files (x86)\Renesas\RL78\1_1_0\
Renesas CCRX	
☑ v2.08.00	C:\Program Files (x86)\Renesas\RX\2_8_0\
☑ v2.07.00	C:\Program Files (x86)\Renesas\RX\2_7_0\
☑ v2.06.00	C:\Program Files (x86)\Renesas\RX\2_6_0\
☑ v2.05.00	C:\Program Files (x86)\Renesas\RX\2_5_0\
☑ v2.04.01	C:\Program Files (x86)\Renesas\RX\2_4_1\
☑ v2.03.00	C:\Program Files (x86)\Renesas\RX\2_3_0\
☑ v2.02.00	C:\Program Files (x86)\Renesas\RX\2_2_0\
Linaro	
KPIT GNURL78-ELF Toolchain	
GCC for Renesas RX	
KPIT GNURX-ELF Toolchain	

Figure 2-7 Toolchain Management

CHAPTER 3. PROJECT GENERATION

This chapter describes the creation of new project and import of existing e^2 studio project, High-performance Embedded Workshop IDE (described as “HEW” below) project and CS+ project to e^2 studio IDE.

Note: 1. To install and use the e^2 studio on your PC, you must install the compiler package provided separately.

2. Multi-byte characters cannot be used for e^2 studio installation folder name, project name and its folder, and source file name.

3.1. New Project Generation

To create a new project with Renesas RXC toolchain, invoke e^2 studio IDE from the Windows ([Start] menu) and specify a workspace directory.

(1) Click [File] → [New] → [C/C++ Project] to open new project creation wizard.

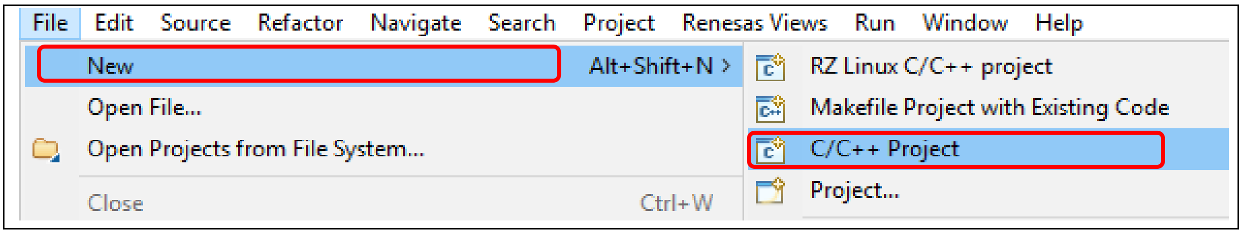
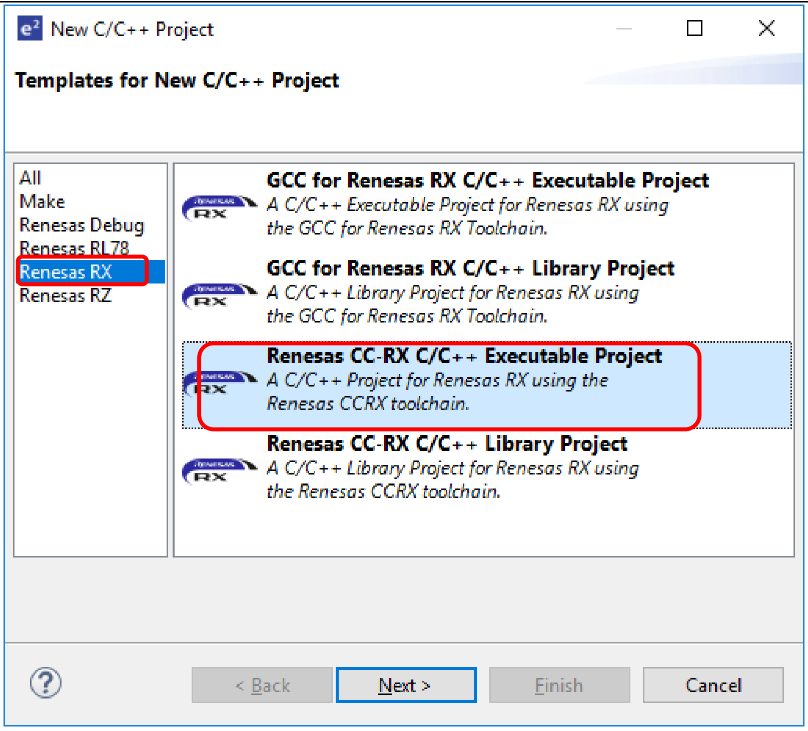


Figure 3-1 Open new project creation wizard

(2) Select template for the new project (For e.g., Renesas RX: “Renesas CC-RX C/C++ Executable Project”). Click [Next] to proceed.



GCC for Renesas RX C/C++ Executable Project
A C/C++ Executable Project for Renesas RX using the GCC for Renesas RX Toolchain.

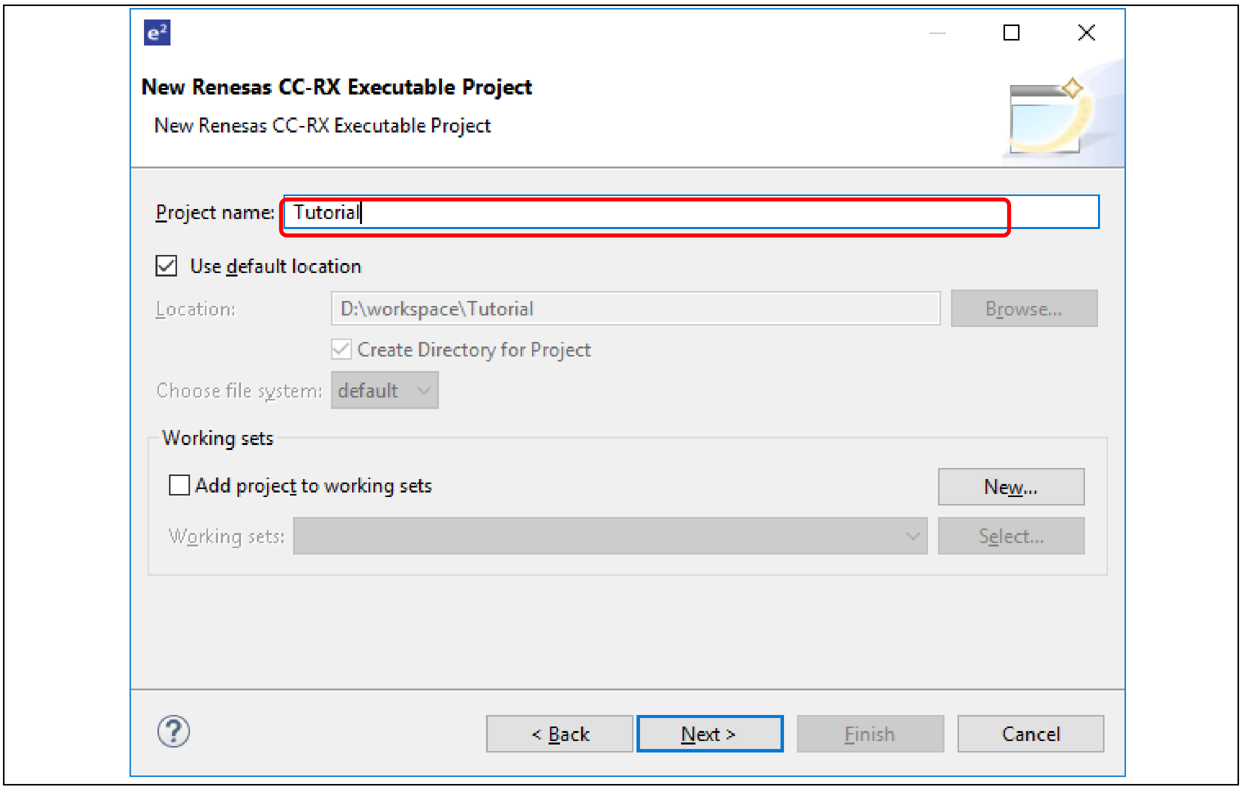
GCC for Renesas RX C/C++ Library Project
A C/C++ Library Project for Renesas RX using the GCC for Renesas RX Toolchain.

Renesas CC-RX C/C++ Executable Project
A C/C++ Project for Renesas RX using the Renesas CCRX toolchain.

Renesas CC-RX C/C++ Library Project
A C/C++ Library Project for Renesas RX using the Renesas CCRX toolchain.

Figure 3-2 New Project Creation Wizard (1/6)

(3) Enter the project name. Click [Next] to proceed.



New Renesas CC-RX Executable Project

New Renesas CC-RX Executable Project

Project name: Tutorial

☑ Use default location

Location: D:\workspace\Tutorial

☑ Create Directory for Project

Choose file system: default

Working sets

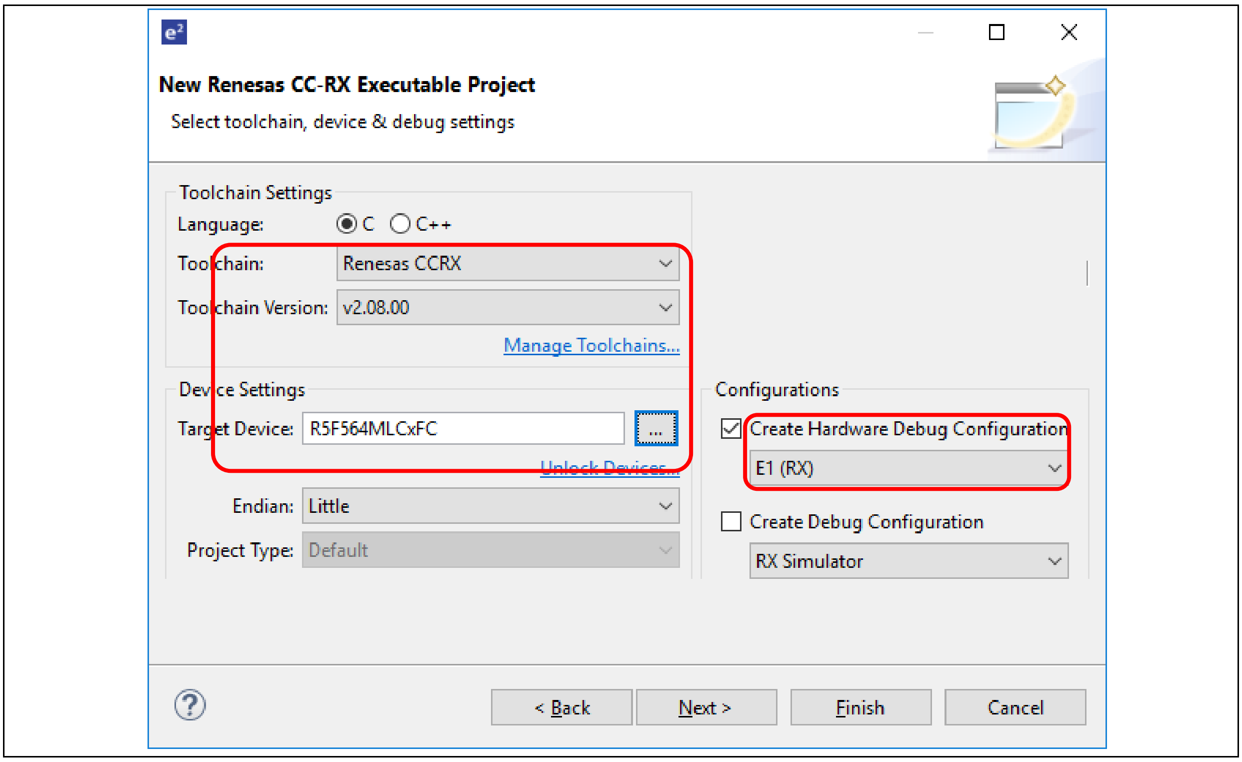
□ Add project to working sets

Working sets:

Figure 3-3 New Project Creation Wizard (2/6)

(4) Select Language, Toolchain, Toolchain Version, Target Device and Configurations. (For e.g., Language: “C”, Toolchain: “Renesas CCRX”, Toolchain Version: “v2.08.00”, Target Device: “R5F564MLCxFC”, Create Hardware Debug Configuration: “E1 (RX)”). Click [Next] to proceed.

Note: “E2 Lite” can be selected in the same way as E1 in the Hardware Debug Configuration pull down menu.



New Renesas CC-RX Executable Project

Select toolchain, device & debug settings

Toolchain Settings

Language: ● C ○ C++

Toolchain: Renesas CCRX

Toolchain Version: v2.08.00

Device Settings

Target Device: R5F564MLCxFC

Endian: Little

Project Type: Default

Configurations

☑ Create Hardware Debug Configuration

E1 (RX)

□ Create Debug Configuration

RX Simulator

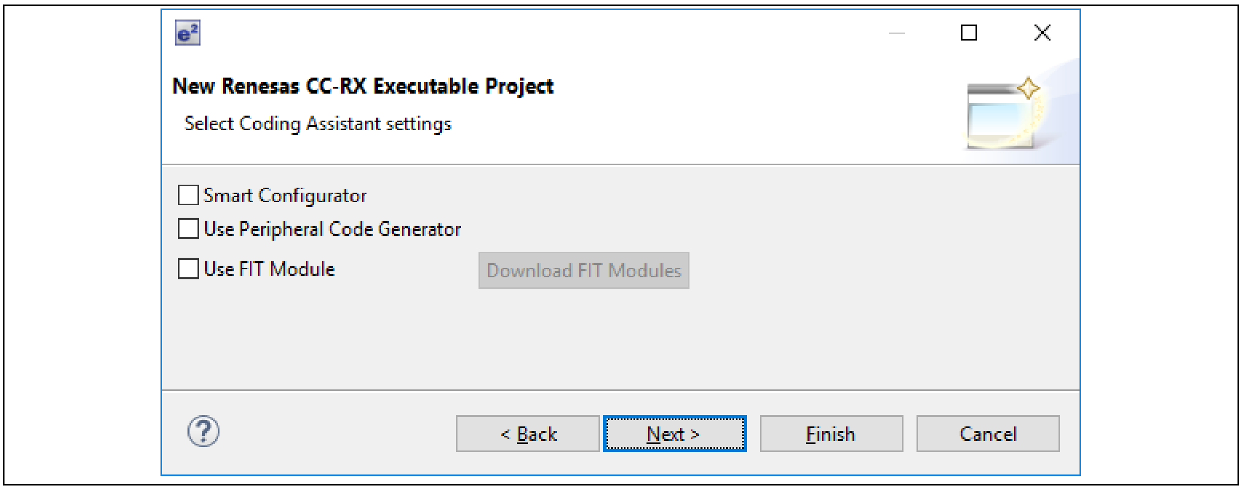
Figure 3-4 New Project Creation Wizard (3/6)

(5) Select the Coding Assistant settings if user know how to use them, otherwise ignore this setting. Click [Next] to proceed.

Note:

- *Peripheral Code Generator* supports the generation of driver and peripheral function code based on GUI settings. Functions are provided as APIs and are not limited to initialization of peripheral function.
- *FIT* not only supports the sample code to be easily embedded into a user application but also provides a common interface between the user applications and peripheral function drivers and middleware.
- *Smart Configurator* supports a single user interface that combines the functionalities of Code Generator and FIT Configurator. Smart Configurator encompasses unified clock configuration view, interrupt configuration view and pin configuration view.

Peripheral Code Generator and Smart Configurator may not be available for some devices.



New Renesas CC-RX Executable Project

Select Coding Assistant settings

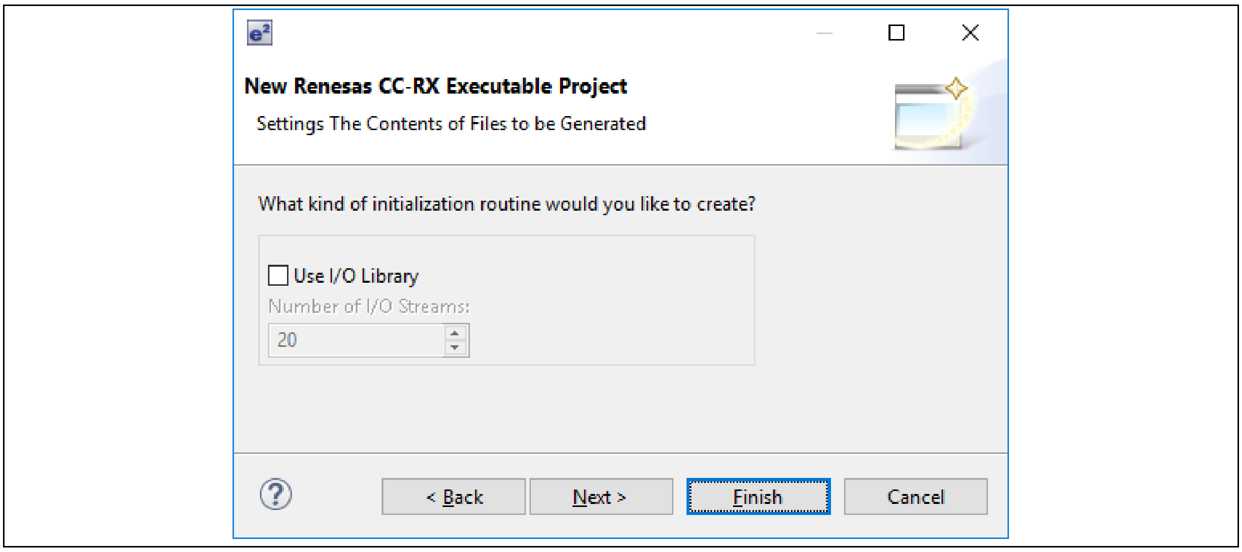
□ Smart Configurator

□ Use Peripheral Code Generator

□ Use FIT Module

Figure 3-5 New Project Creation Wizard (4/6)

(6) Keep the “Use I/O Library” unchecked and click [Next] to proceed.



New Renesas CC-RX Executable Project

Settings The Contents of Files to be Generated

What kind of initialization routine would you like to create?

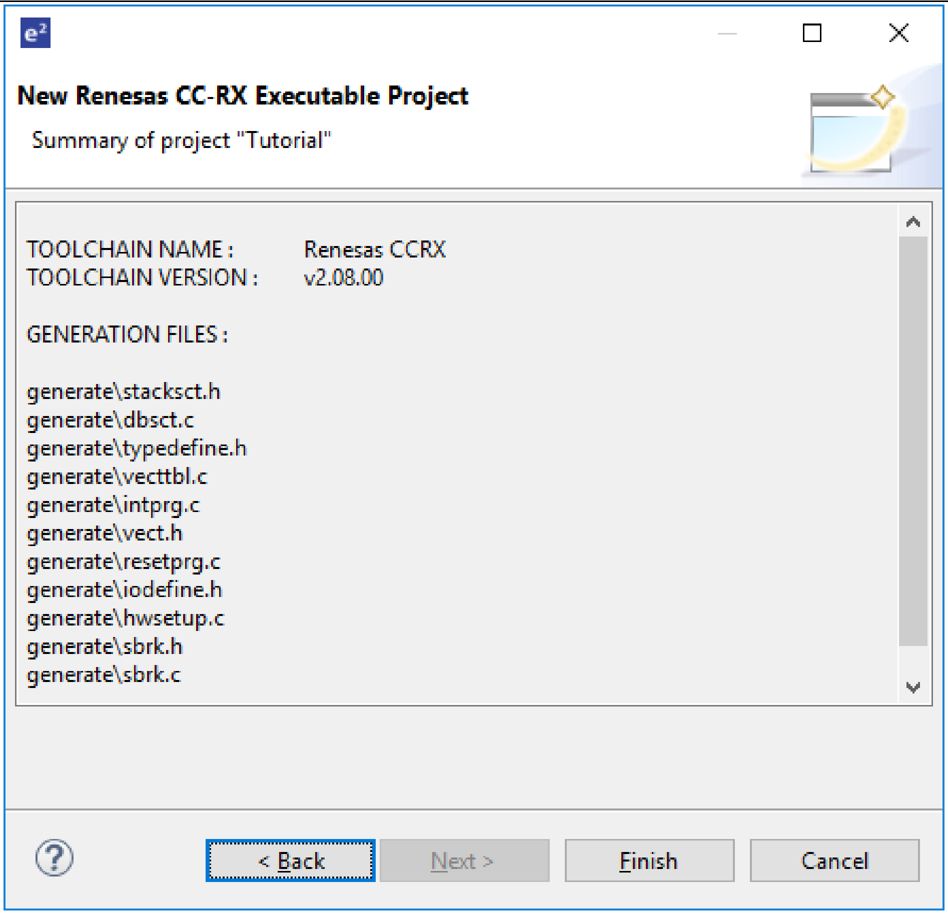
□ Use I/O Library

Number of I/O Streams:

20

Figure 3-6 New Project Creation Wizard (5/6)

(7) A project summary is displayed. Click [Finish] to generate the project.



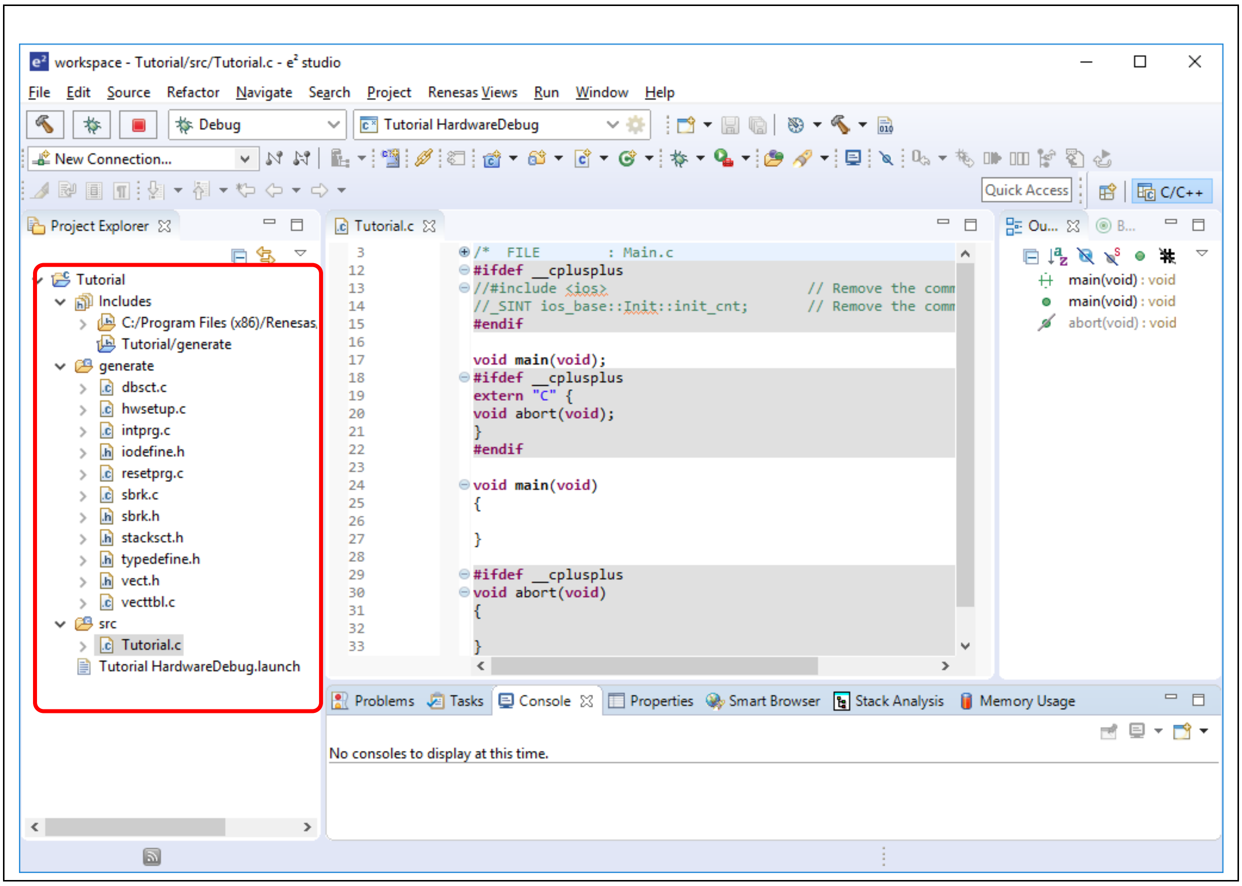
New Renesas CC-RX Executable Project

Summary of project "Tutorial"

```
TOOLCHAIN NAME :        Renesas CCRX
TOOLCHAIN VERSION :     v2.08.00

GENERATION FILES :

generate\stacksct.h
generate\dbsct.c
generate\typedefine.h
generate\vecttbl.c
generate\intprg.c
generate\vect.h
generate\resetprg.c
generate\iodefine.h
generate\hwsetup.c
generate\sbrk.h
generate\sbrk.c
```

Figure 3-7 New Project Creation Wizard (6/6)

(8) A new C project named “Tutorial” is created.



```
/*  FILE        : Main.c
#ifdef __cplusplus
//#include <ios>                        // Remove the comm
//_SINT ios_base::Init::init_cnt;       // Remove the comm
#endif

void main(void);
#ifdef __cplusplus
extern "C" {
void abort(void);
}
#endif

void main(void)
{

}

#ifdef __cplusplus
void abort(void)
{

}
```

No consoles to display at this time.

Figure 3-8 New C Project Created

This project consists of an application file “Tutorial.c” and standard start-up files (e.g. “dbsct.c”, “intprg.c”, “sbrk.c” etc). All these project and source files listed in the [Project Explorer] panel reflect the folder structure of the project, just as seen on the standard file explorer.

Notes for backing up projects:

- Project properties are stored in files or folders which filenames or folder names are prefixed with a '.' (dot), for example ".project" and ".cproject". It is necessary to include these files or folders when archiving the project for back-up purpose.
- In order to restore properties shared among projects, for instance when one project makes reference to another project's files, please backup the whole workspace folder.

3.2. Import Existing Projects Into Workspace

The migration guideline between integrated development environments can be found at the following site.

https://www.renesas.com/products/software-tools/tools/migration-tools/migration-tools-ide.html

CHAPTER 4. BUILD

This chapter describes the build configurations and key build features for e^2 studio IDE.

4.1. Build Option Settings

The default build option is generated when a project is created and it can usually be used to build the project. However, if changing build option is necessary (e.g. Toolchain version, Optimization options, etc.), follow the following steps before building the project.

(1) Right click on project “Tutorial” and select [Properties] or use shortcut keys [Alt] + [Enter] to open the Properties window.

Properties window is supported at workspace, project and source level. Properties window for project supports more configurations which apply across all the files within the same project workspace.

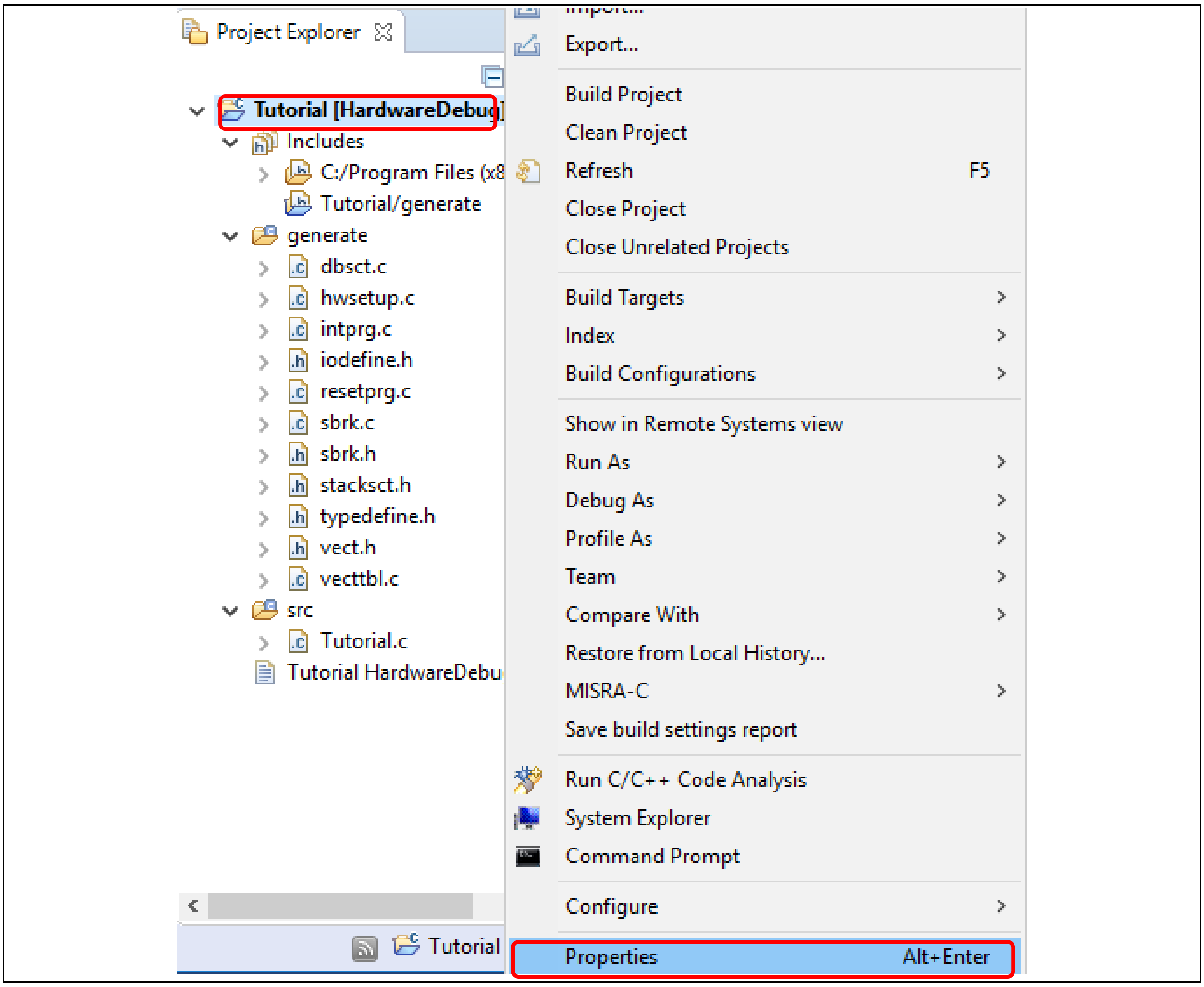
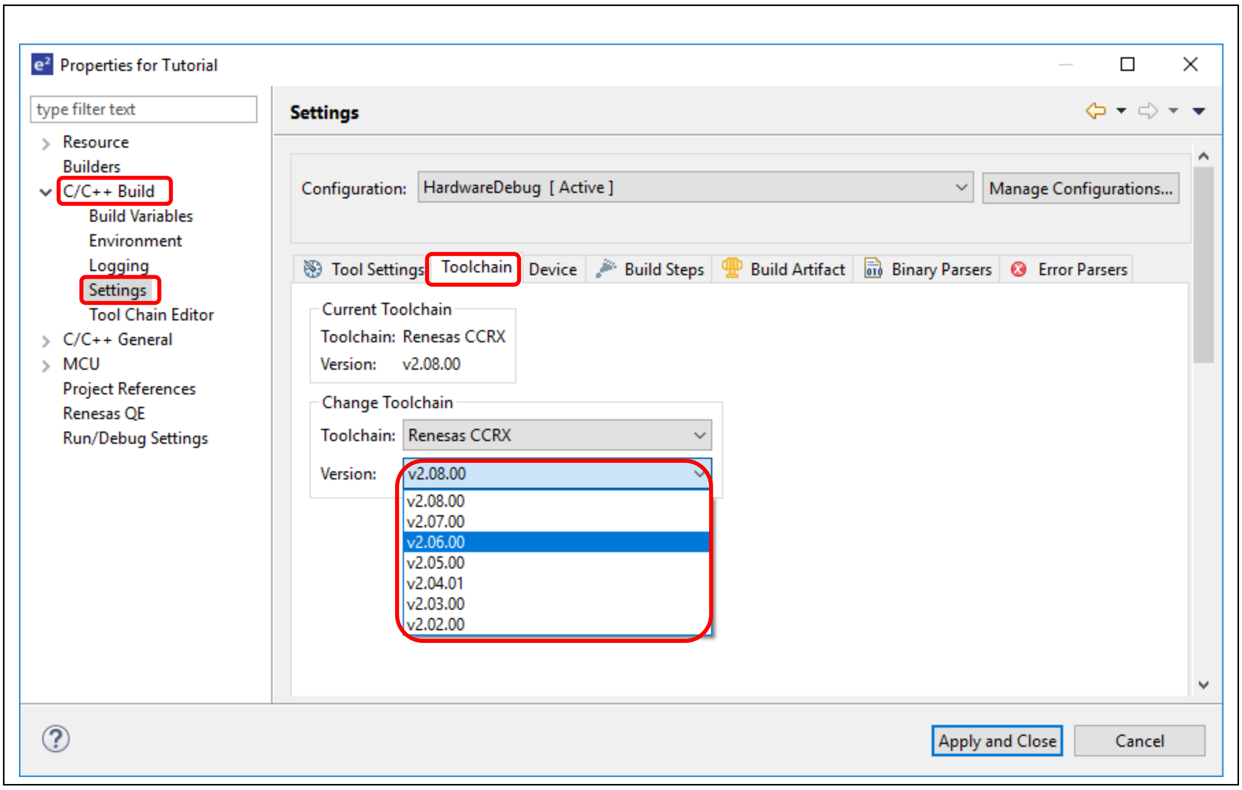


Figure 4-1 Open the Properties window

(2) Click [C/C++ Build] → [Change Toolchain Version] to view or change toolchain version.
Refer to figure 4-2, the current version is v2.08.00 and click the “Versions” option to change toolchain version (if additional toolchain is installed).



Configuration: HardwareDebug [Active]

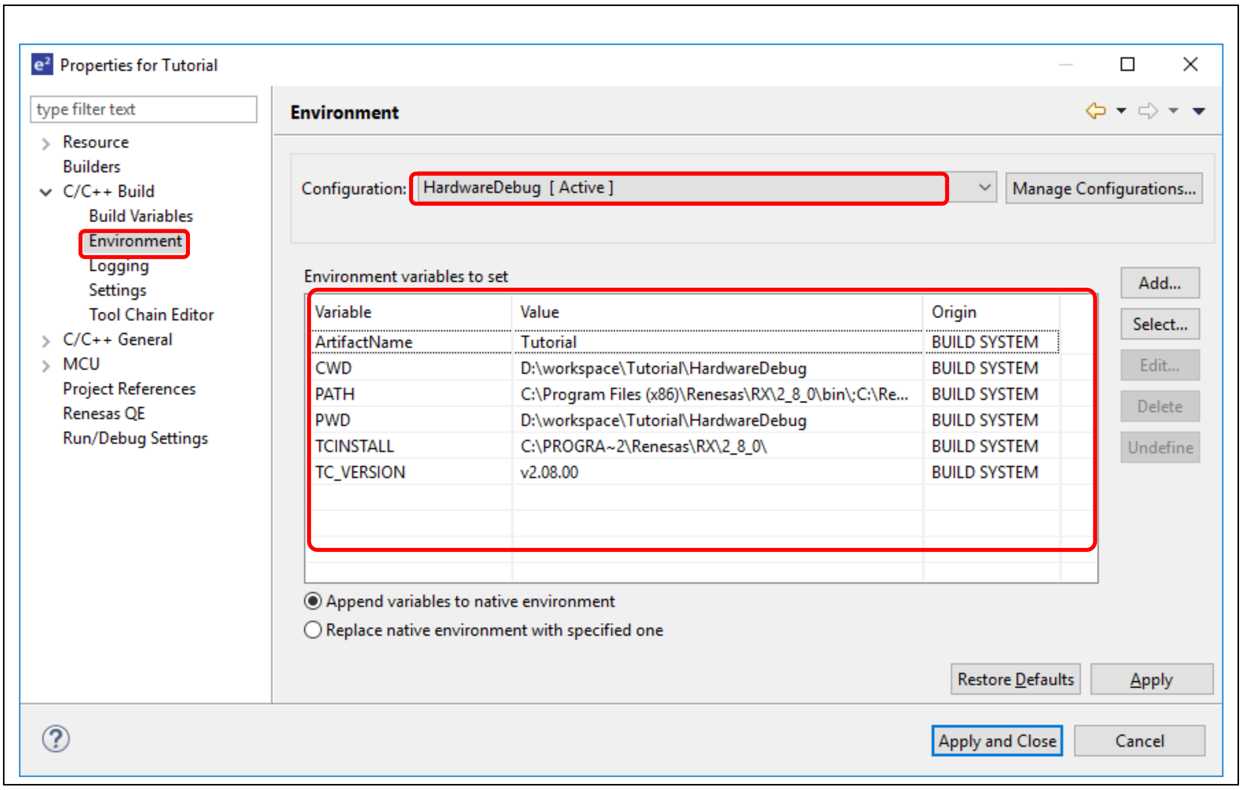
Current Toolchain
Toolchain: Renesas CCRX
Version: v2.08.00

Change Toolchain
Toolchain: Renesas CCRX
Version: v2.08.00

v2.08.00
v2.07.00
v2.06.00
v2.05.00
v2.04.01
v2.03.00
v2.02.00

Figure 4-2 Change Toolchain Version

(3) Click [C/C++ Build] → [Environment] to set build option and add or edit the environment variables.



Configuration: HardwareDebug [Active]

Environment variables to set

Variable	Value	Origin
ArtifactName	Tutorial	BUILD SYSTEM
CWD	D:\workspace\Tutorial\HardwareDebug	BUILD SYSTEM
PATH	C:\Program Files (x86)\Renesas\RX\2_8_0\bin\;C:\Re...	BUILD SYSTEM
PWD	D:\workspace\Tutorial\HardwareDebug	BUILD SYSTEM
TCINSTALL	C:\PROGRA~2\Renesas\RX\2_8_0\	BUILD SYSTEM
TC_VERSION	v2.08.00	BUILD SYSTEM

● Append variables to native environment
○ Replace native environment with specified one

Figure 4-3 Build Settings for Compiler: Environment

Build option allows user to retain all the toolchain configuration settings, including path name specified by using the environment variables. The current build configuration is “HardwareDebug [Active]”, as shown in Figure 4-3.

The detail of build option is described in compiler user manual which is stored at “{Compiler installation directory}\doc”. For example, it can be found in “C:\Program Files\Renesas\RX\2_8_0\doc\ ".

4.2. Build A Sample Project

A project can be built by one of the ways below:

(1) Right click on the project and select [Build Project]

(2) Click on the project to set focus and select [Project] → [Build Project]

(3) Click on the project to set focus and click on 🔨 icon.

(4) Click on the project to set focus and press [Ctrl] + [B]

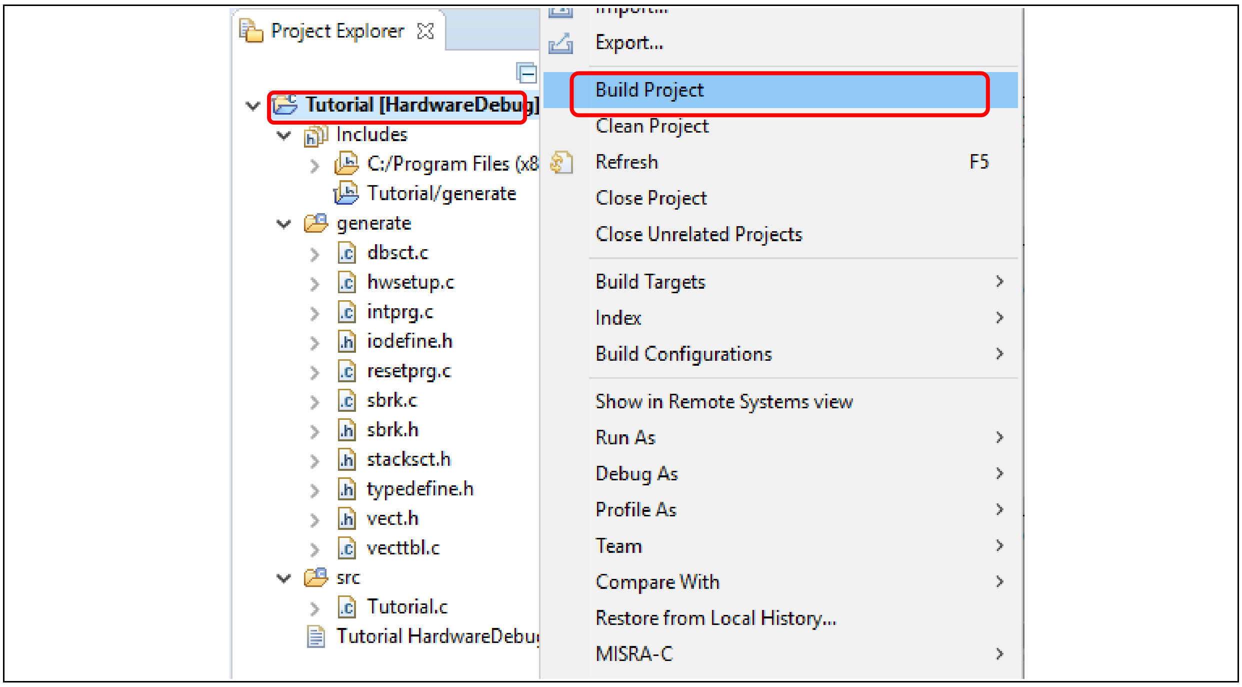
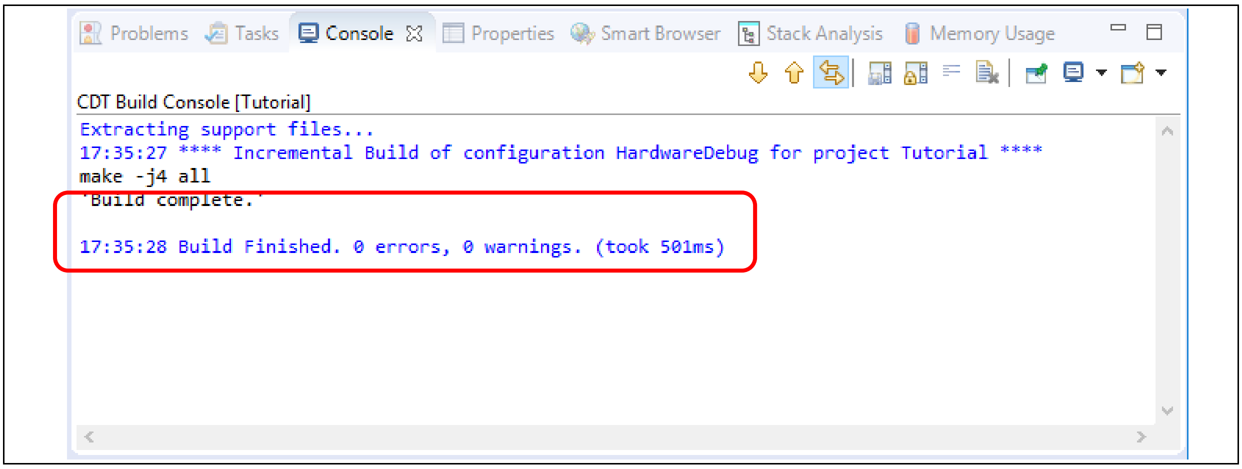


Figure 4-4 Build a Sample "Tutorial" Project

The [Console] pane shows 'Build complete.' message to indicate a successful build. At the end of this build, files output to the ${CONFIGDIR} directory consists of "makefile", "Tutorial.abs", "Tutorial.map", "Tutorial.mot", "Tutorial.x" etc.

"Tutorial.abs" is a Renesas standard load module in ELF/DWARF format (*.abs) used for the debugging. Because GDB supports a load module format with different ELF/DWARF specification (*.x or *.elf), hence "Tutorial.abs" has to be converted to "Tutorial.x" for the debugging in e^2 studio IDE.



CDT Build Console [Tutorial]

```
Extracting support files...
17:35:27 **** Incremental Build of configuration HardwareDebug for project Tutorial ****
make -j4 all
'Build complete.'

17:35:28 Build Finished. 0 errors, 0 warnings. (took 501ms)
```

Figure 4-5 Project is built successfully.

4.3. Export Build Configuration Settings

The Project Reporter feature can export project and build configuration settings from e^2 studio IDE to a file for easy checking and comparison of project/build environment settings.

(1) Right-click at [Project Explorer] to pop up context menu

(2) Select [Save build settings report] to save build settings report

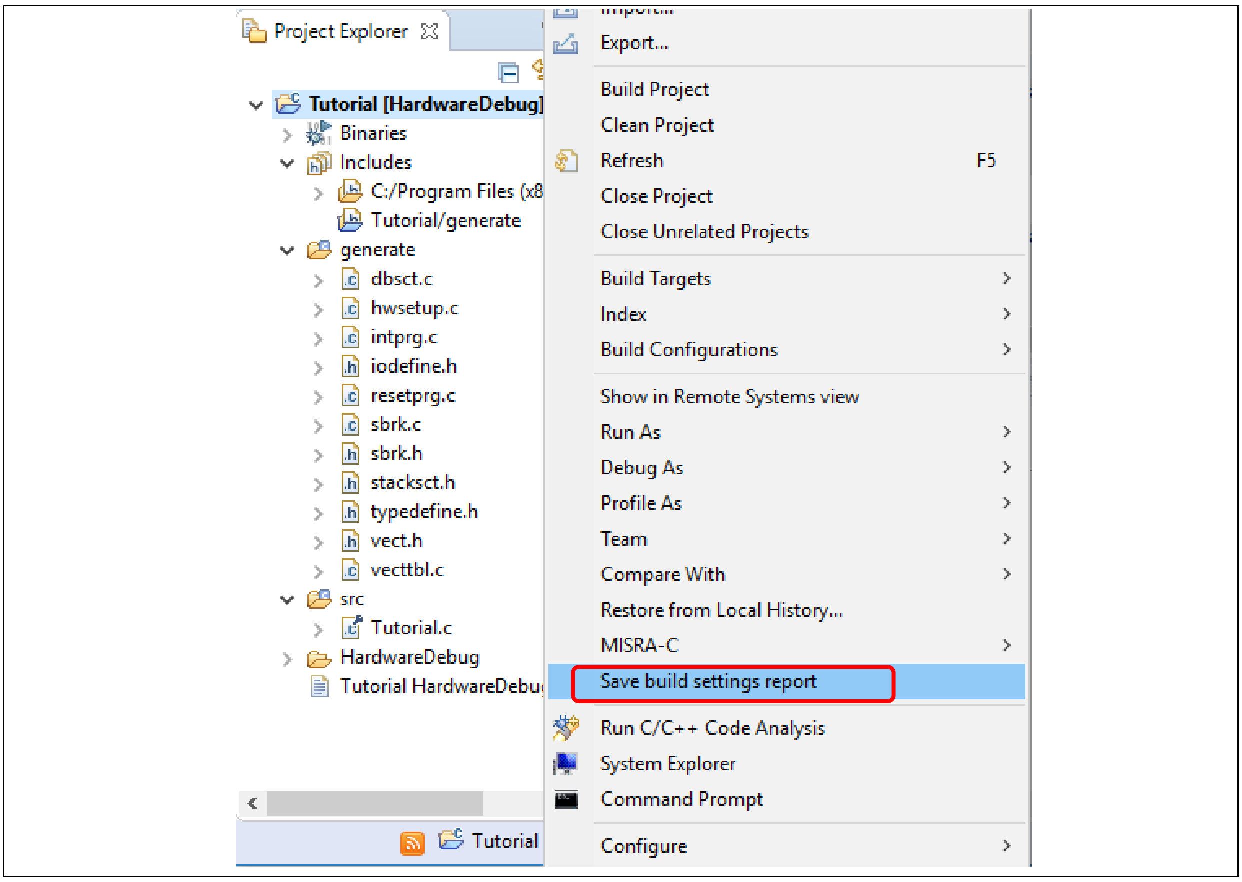


Figure 4-6 Project Reporter

CHAPTER 5. DEBUG

This chapter describes the usage of debug configuration and key debugging features for e^2 studio IDE. The following illustration refers to “Tutorial” project built (in Chapter 4.2) and based on hardware configuration: E1 emulator or E2 emulator Lite and RSK RX64M board.

(1) Open “Tutorial” project workspace in e^2 studio IDE and click [Debug] perspective.

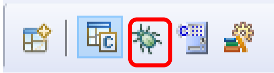


Figure 5-1 Switch to [Debug] Perspective

Perspective defines the layout views (related to development tools) in the Workbench window. Each perspective consists of a combination of views, menus and toolbars that enable user to perform specific task.

For instance, [C/C++] perspective has views that help user to develop C/C++ programs and [Debug] perspective has views that enable user to debug the program. If user attempts to connect up the debugger in the [C/C++] perspective, IDE will then prompt users to switch to the [Debug] perspective.

One or more perspectives can exist in a single Workbench window. User can customize them or add new perspective.

Note: For more information on debug, please refer to “e^2 studio Debug Help” as described in chapter 6.

5.1. Change Existing Debug Configurations

The debug configuration has to be configured when debugging for the first time and it just needs to be done once. An existing debug configuration can be changed as follows.

(1) Click “Tutorial” Project in [Project Explorer] pane to set focus.

Click [Run] → [Debug Configurations…] or icon (downward arrow) → [Debug Configurations…] to open the “Debug Configurations” window.

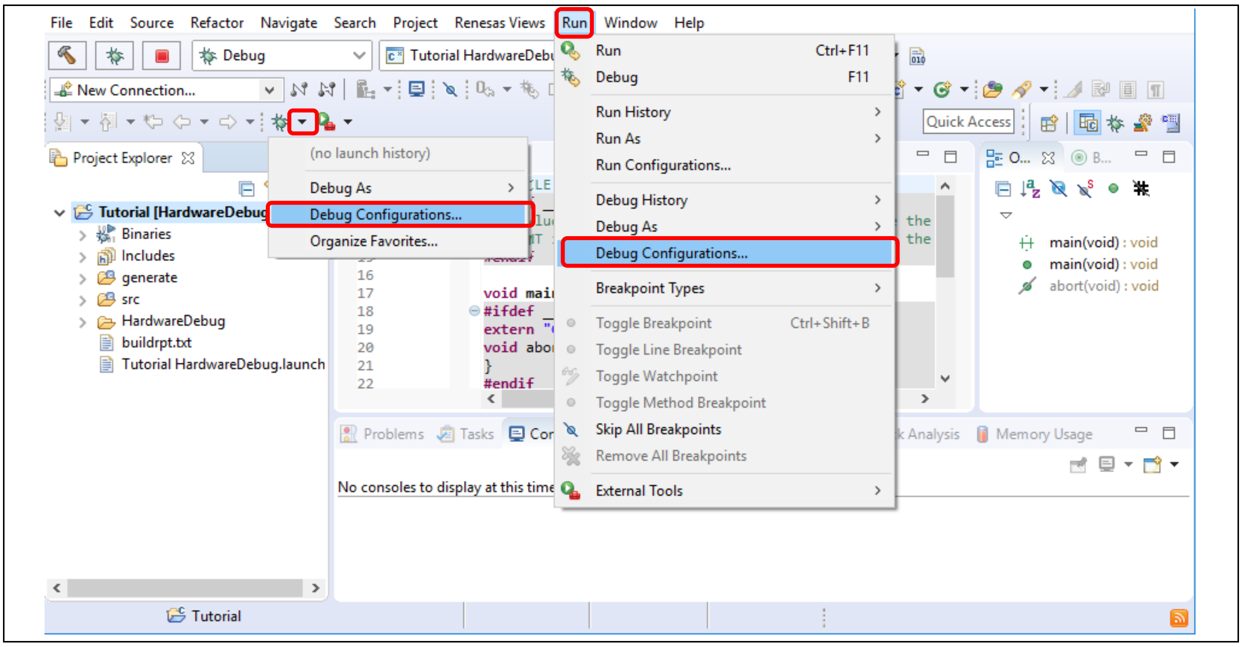
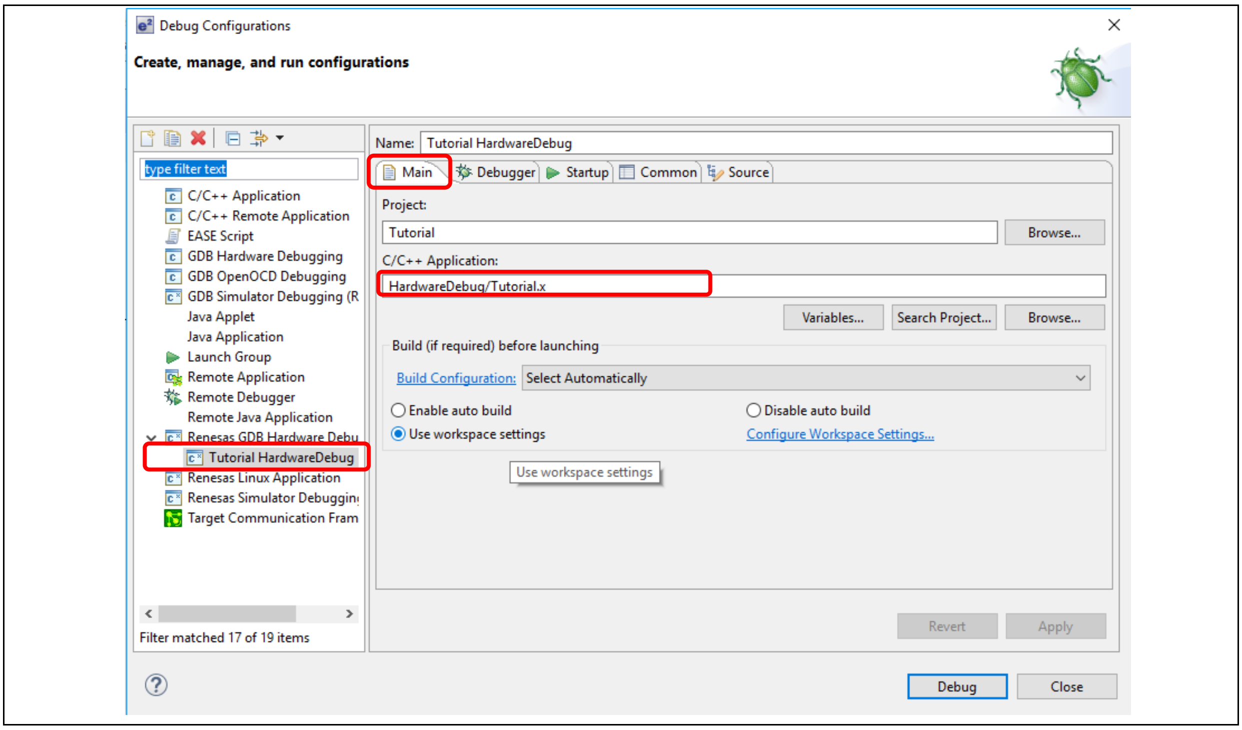


Figure 5-2 Open Debug Configurations Window

(2) In “Debug Configurations” windows, go to [Renesas GDB Hardware Debugging] → [Tutorial HardwareDebug]. Click on the [Main] tab to ensure the load module is “Tutorial.x”.


Debug Configurations

Create, manage, and run configurations

Name: Tutorial HardwareDebug

Project: Tutorial

C/C++ Application: HardwareDebug/Tutorial.x

Build (if required) before launching

Build Configuration: Select Automatically

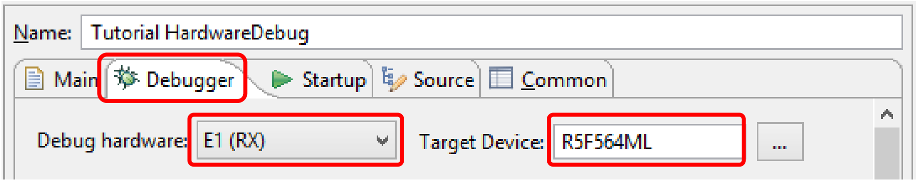
○ Enable auto build ○ Disable auto build

● Use workspace settings

Filter matched 17 of 19 items

Figure 5-3 Select Load Module

(3) Switch to the [Debugger] tab, set “E1” as the debug hardware and “R5F564ML” as the target device.



Name: Tutorial HardwareDebug

Debug hardware: E1 (RX) Target Device: R5F564ML

Figure 5-4 Select Target Device

(4) Under the [Debugger] tab, go to the [Connection Settings] sub tab to configure the following based on the settings in E1 emulator and RSK RX64M board:

- Clock
  - Main Clock Source = “EXTAL”
  - Extal Frequency(MHz) = “24.0000”

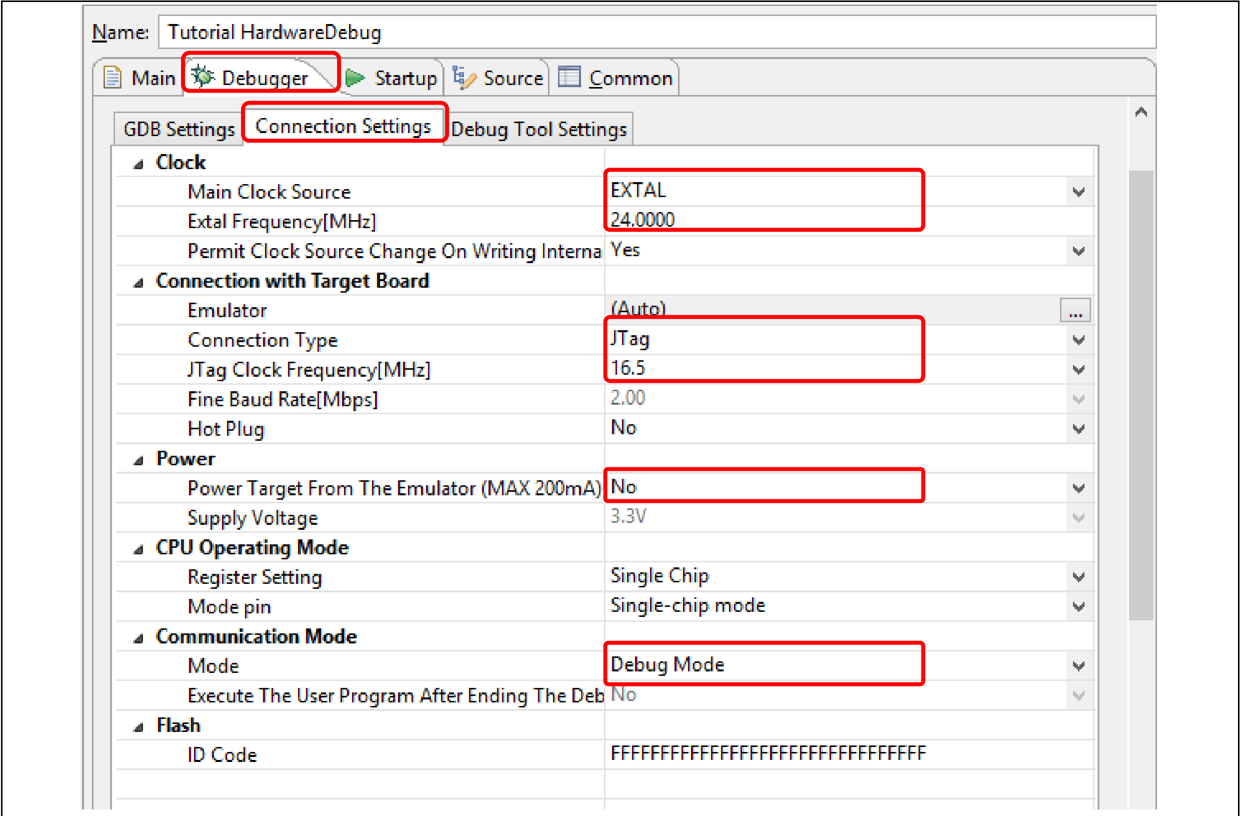
  Note: Extal frequency is the value printed on the oscillator device on your board.
- Connection with Target Board

  - Connection Type = “JTag”

  - JTag Clock Frequency [MHz] = “16.5”
- Power
  - Power Target From The Emulator (MAX 200mA) = “No”
- Communication Mode
  - Mode = “Debug Mode”

When “Power Target From The Emulator (MAX 200mA)” is set to “Yes”, the emulator will power up (with current up to 200mA) the target board without an external power source.

Note: This debug configuration in Figure 5-5 is shown as an example. The wrong settings may cause malfunction or damage to the hardware. So, be cautious to verify the board and emulator settings before connection.



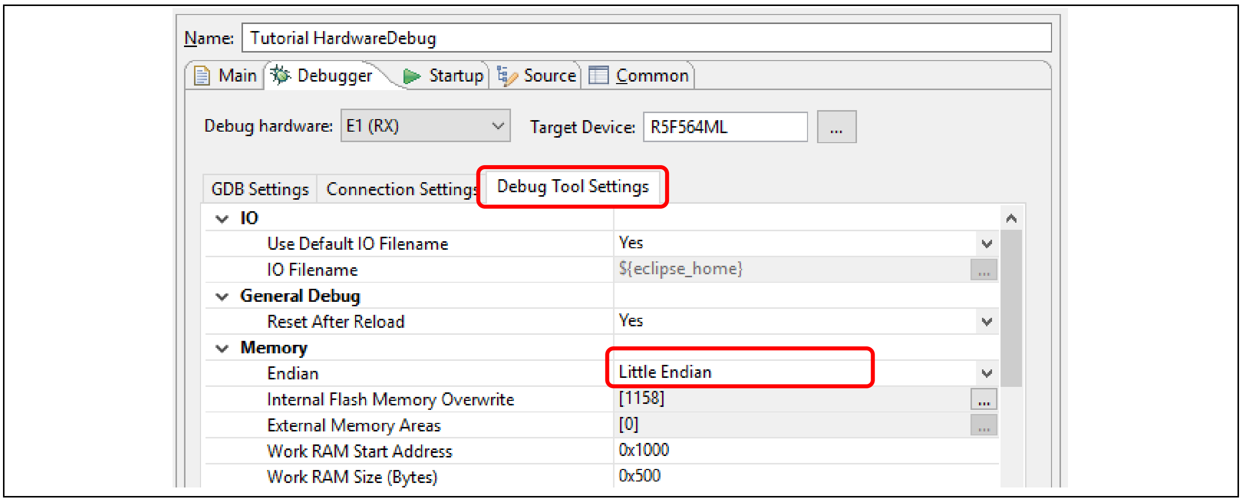
Name: Tutorial HardwareDebug

Clock	
Main Clock Source	EXTAL
Extal Frequency[MHz]	24.0000
Permit Clock Source Change On Writing Interna	Yes
Connection with Target Board	
Emulator	(Auto)
Connection Type	JTag
JTag Clock Frequency[MHz]	16.5
Fine Baud Rate[Mbps]	2.00
Hot Plug	No
Power	
Power Target From The Emulator (MAX 200mA)	No
Supply Voltage	3.3V
CPU Operating Mode	
Register Setting	Single Chip
Mode pin	Single-chip mode
Communication Mode	
Mode	Debug Mode
Execute The User Program After Ending The Deb	No
Flash	
ID Code	FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

Figure 5-5 Change Connection Setting

(5) Switch to [Debug Tool Settings] sub tab, based on the RSK RX64M board to ensure

- Memory
  Endian = “Little Endian”



Name: Tutorial HardwareDebug

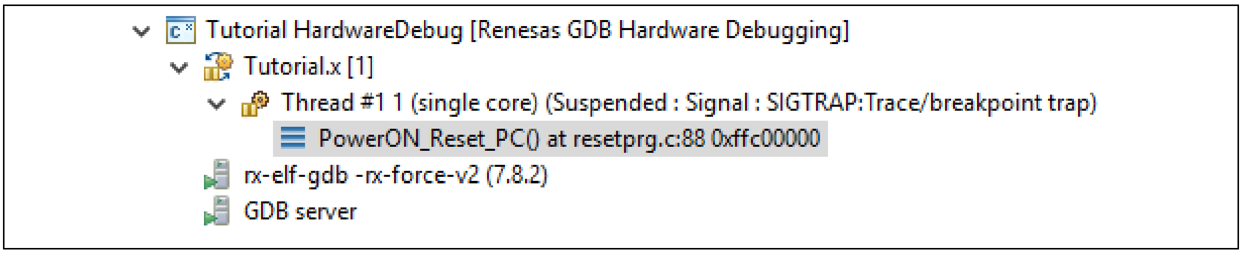
Debug hardware: E1 (RX) Target Device: R5F564ML

IO	
Use Default IO Filename	Yes
IO Filename	${eclipse_home}
General Debug	
Reset After Reload	Yes
Memory	
Endian	Little Endian
Internal Flash Memory Overwrite	[1158]
External Memory Areas	[0]
Work RAM Start Address	0x1000
Work RAM Size (Bytes)	0x500

Figure 5-6 Change Debug Tool Settings

(6) Click [Apply] button to confirm the settings. Then click [Debug] to execute the debug launch configuration to connect to the E1 (or E2 Lite) and RSK RX64M board.

(7) For a successful connection, [Debug] view to show target debugging information in a tree hierarchy. The program entry point is set at “PowerON_Reset() in “resetprg.c”.



Tutorial HardwareDebug [Renesas GDB Hardware Debugging]
Tutorial.x [1]
Thread #1 1 (single core) (Suspended : Signal : SIGTRAP:Trace/breakpoint trap)
PowerON_Reset_PC() at resetprg.c:88 0xffc00000
rx-elf-gdb -rx-force-v2 (7.8.2)
GDB server

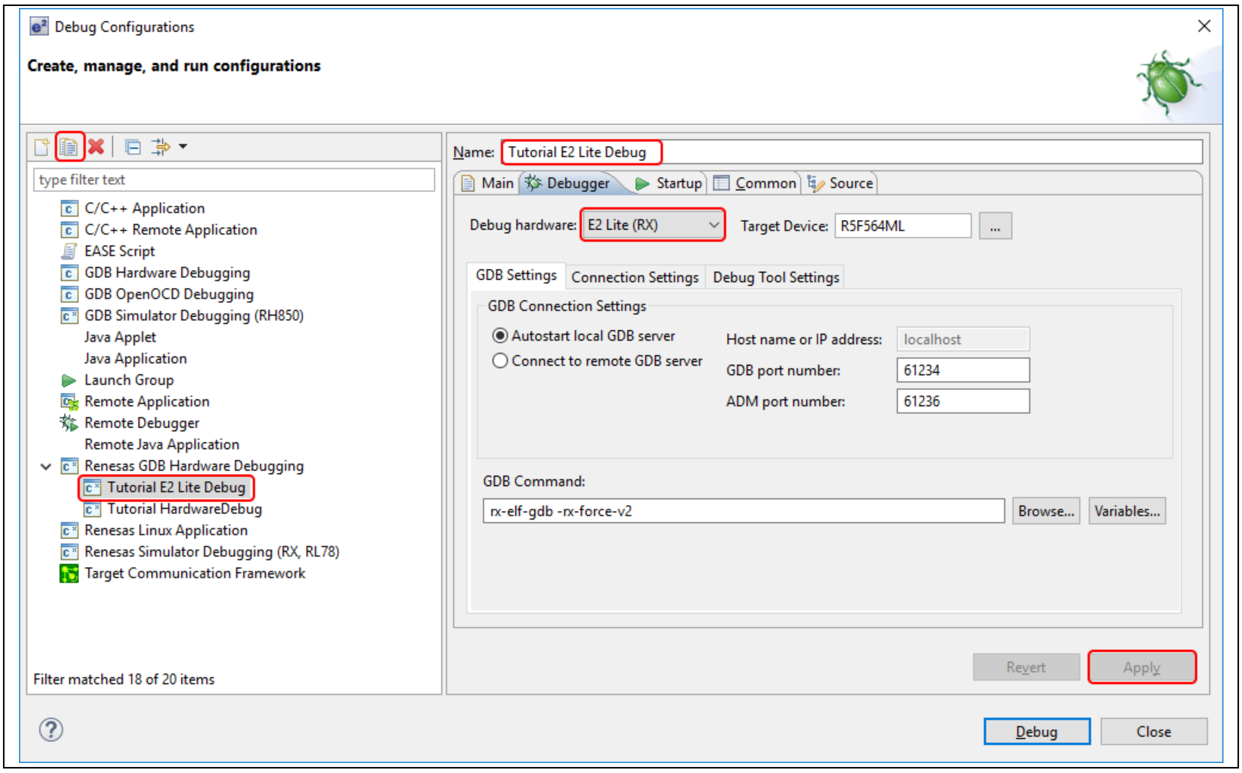
Figure 5-7 User Target Connection in the [Debug] View

5.2. Create New Debug Configurations

The simplest way to create a new debug configuration is by duplicating an existing one. It can be done by the following steps.

(1) Repeat step 1 in section 5.1 to open “Debug Configurations” window.

(2) Select a debug configuration (e.g. “Tutorial HardwareDebug”) and then click icon (Duplicates the currently selected launch configuration). A new debug launch configuration (e.g. “Tutorial HardwareDebug (1)”) is created. User can rename it to identify the settings by typing in the “Name” textbox then click [Apply] button.



Debug Configurations

Create, manage, and run configurations

type filter text

- C/C++ Application
- C/C++ Remote Application
- EASE Script
- GDB Hardware Debugging
- GDB OpenOCD Debugging
- GDB Simulator Debugging (RH850)
- Java Applet
- Java Application
- Launch Group
- Remote Application
- Remote Debugger
- Remote Java Application
- Renesas GDB Hardware Debugging
  - Tutorial E2 Lite Debug
  - Tutorial HardwareDebug
- Renesas Linux Application
- Renesas Simulator Debugging (RX, RL78)
- Target Communication Framework

Filter matched 18 of 20 items

Name: Tutorial E2 Lite Debug

Main | Debugger | Startup | Common | Source

Debug hardware: E2 Lite (RX) Target Device: R5F564ML

GDB Settings | Connection Settings | Debug Tool Settings

GDB Connection Settings

● Autostart local GDB server
○ Connect to remote GDB server

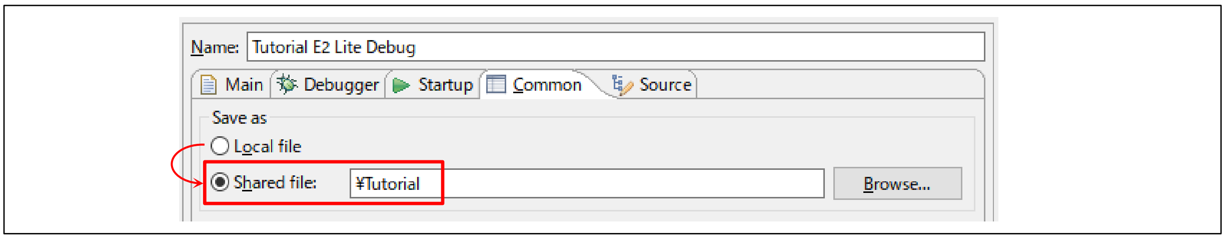
Host name or IP address: localhost
GDB port number: 61234
ADM port number: 61236

GDB Command:
rx-elf-gdb -rx-force-v2

Figure 5-8 Duplicate A Selected Debug Launch Configuration

(1) The debug launch configuration can be configured as described in chapter 5.1. For example, change the Debug Hardware to “E2 Lite (RX)”.

(2) If the launch configuration was added with [local] and * (red star) marker, it isn’t yet attached to any project. Then please specify project name in the Common tab.



Name: Tutorial E2 Lite Debug

Main | Debugger | Startup | Common | Source

Save as
○ Local file
● Shared file: ¥Tutorial

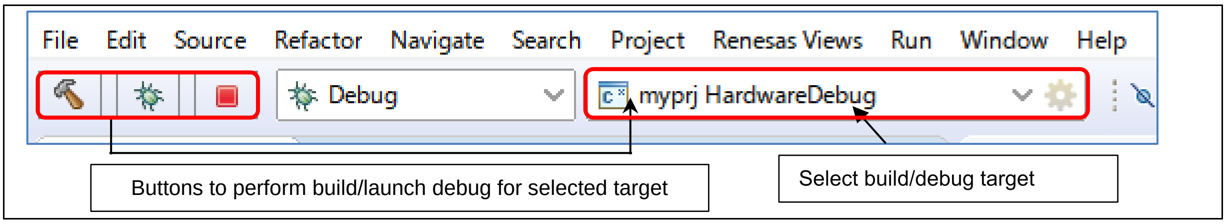
Figure 5-9 Attach Launch Configuration to Specific Project

Notes for RL78 debugging:

- Hot Plug connection is supported for RL78/F1A, F13, F14 and F15 only.

5.3. Launch Bar

This section explains the usage of ‘Launch Bar’, which is supported from V6.0.0 or later version.
Launch Bar is located in the toolbar area of e^2 studio main window.
Interface is very simple as shown below to build and debug for the selected launch target.



Buttons to perform build/launch debug for selected target

Select build/debug target

Figure 5-10 Launch Bar interface

Launch Bar buttons behave as follows:

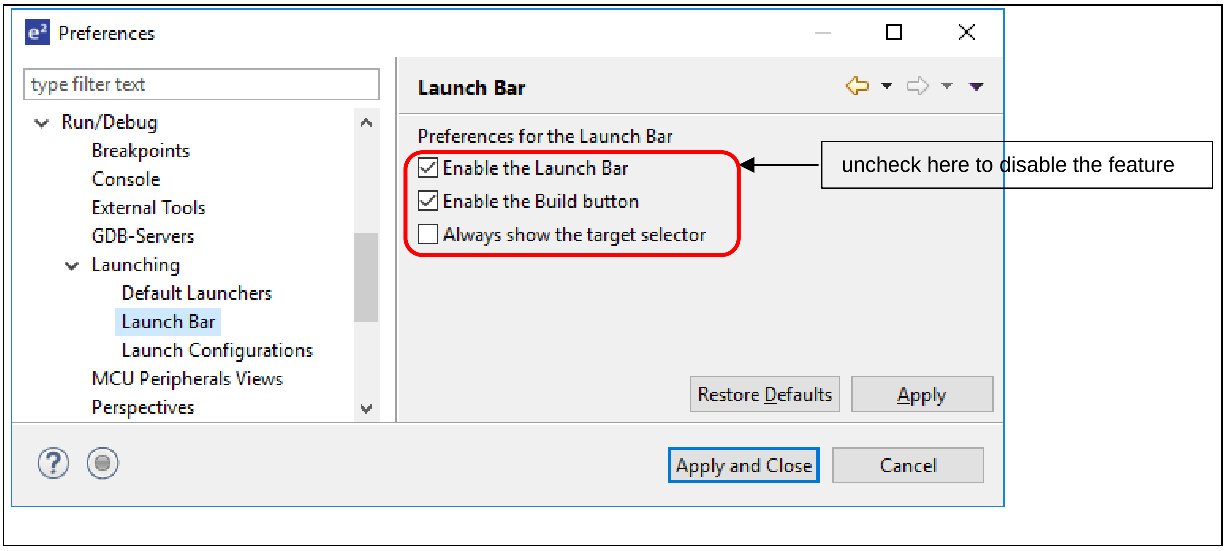
- button builds the load module of the selected launch configuration.

  Note: There is another build button in the project management toolbar builds active build configuration of Project Explorer, while the launch bar does not reflect the active state in Project Explorer.

- buttons are trigger of debugger launch and terminate the selected launch target.

Launch Bar and build button can be hidden through the following dialog.

- Click [Window] menu → [Preferences], then click [Run/Debug] → [Launching] → [Launch Bar]



Launch Bar

Preferences for the Launch Bar

☑ Enable the Launch Bar
☑ Enable the Build button
☐ Always show the target selector

uncheck here to disable the feature

Figure 5-11 Preferences for the Launch Bar

5.4. Basic Debugging Features

This section explains the typical Debug views supported in e^2 studio IDE.

- Standard GDB Debug (supported by Eclipse IDE framework): Breakpoints, Expressions, Registers, Memory, Disassembly and Variables
- Renesas Extension to Standard GDB Debug: Eventpoints, IO Registers and Trace.

The following are some useful buttons exist in the [Debug] view:

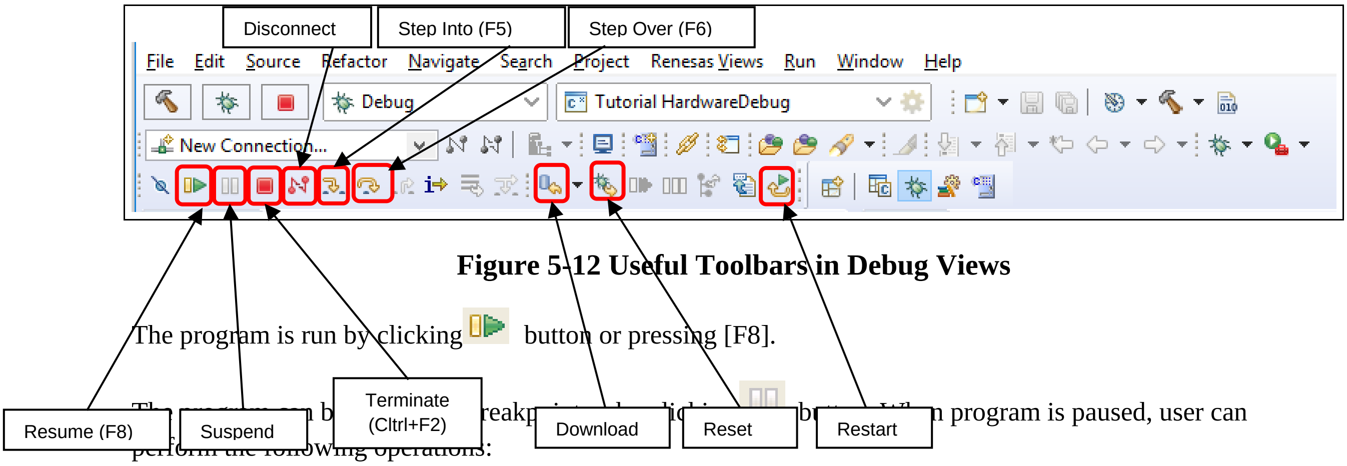


Figure 5-12 Useful Toolbars in Debug Views

The program is run by clicking ▶ button or pressing [F8].

The program can b[illegible] breakp[illegible] clicking ⏸ button. When program is paused, user can perform the following operations:

- ⤵ button or [F5] can be used for stepping into the next method call at the currently executing line of code.
- ↷ button or [F6] can be used for stepping over the next method call (executing but without entering it) at the currently executing line of code.
- ▶ button can be clicked again to resume running.

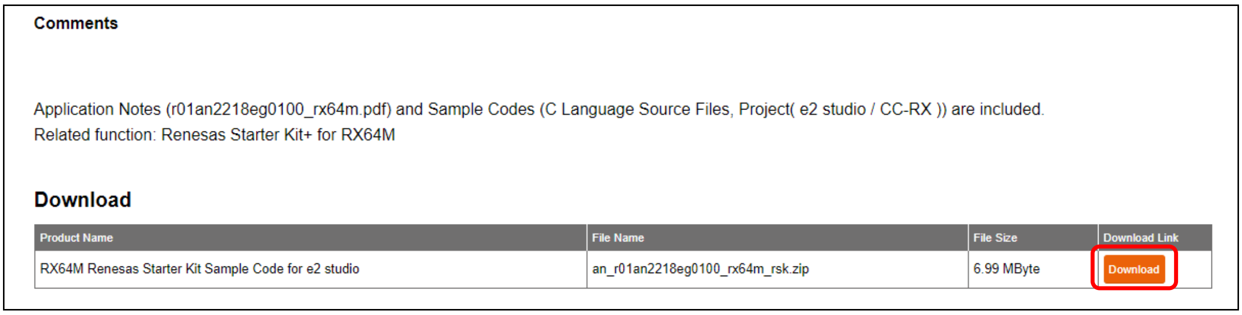
To stop the debugging process, ■ button is clicked to end the selected debug session and/or process or ⛓ button is clicked to disconnect the debugger from the selected process.

The other operations are as following:

- ↻ button can be clicked to start new debug session.
- ⟲ button can be clicked to reset the program to entry point at the PowerOn Reset.
- ⇩ button is used for re-downloading the binary file to target system.

Note: To demonstate the features in following section, please use the sample code for RX64M from Renesas website as follows:

(1) Download the sample code for RX64M from Renesas website: https://www.renesas.com/search/keyword-search.html#q=r01an2218&genre=sampleprogram.



Comments

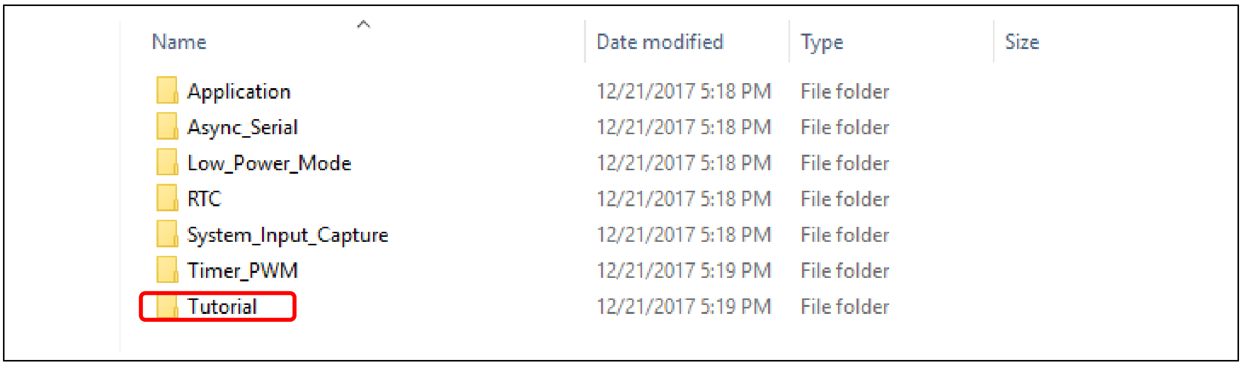
Application Notes (r01an2218eg0100_rx64m.pdf) and Sample Codes (C Language Source Files, Project(e2 studio / CC-RX)) are included.
Related function: Renesas Starter Kit+ for RX64M

Download

Product Name	File Name	File Size	Download Link
RX64M Renesas Starter Kit Sample Code for e2 studio	an_r01an2218eg0100_rx64m_rsk.zip	6.99 MByte	Download

Figure 5-13 Download the Sample Code

(2) After extract the package, find the project "Tutorial"



Name	Date modified	Type	Size
Application	12/21/2017 5:18 PM	File folder	
Async_Serial	12/21/2017 5:18 PM	File folder	
Low_Power_Mode	12/21/2017 5:18 PM	File folder	
RTC	12/21/2017 5:18 PM	File folder	
System_Input_Capture	12/21/2017 5:18 PM	File folder	
Timer_PWM	12/21/2017 5:19 PM	File folder	
Tutorial	12/21/2017 5:19 PM	File folder	

Figure 5-14 The Sample project

(3) In e^2 studio, select [File] → [Import]

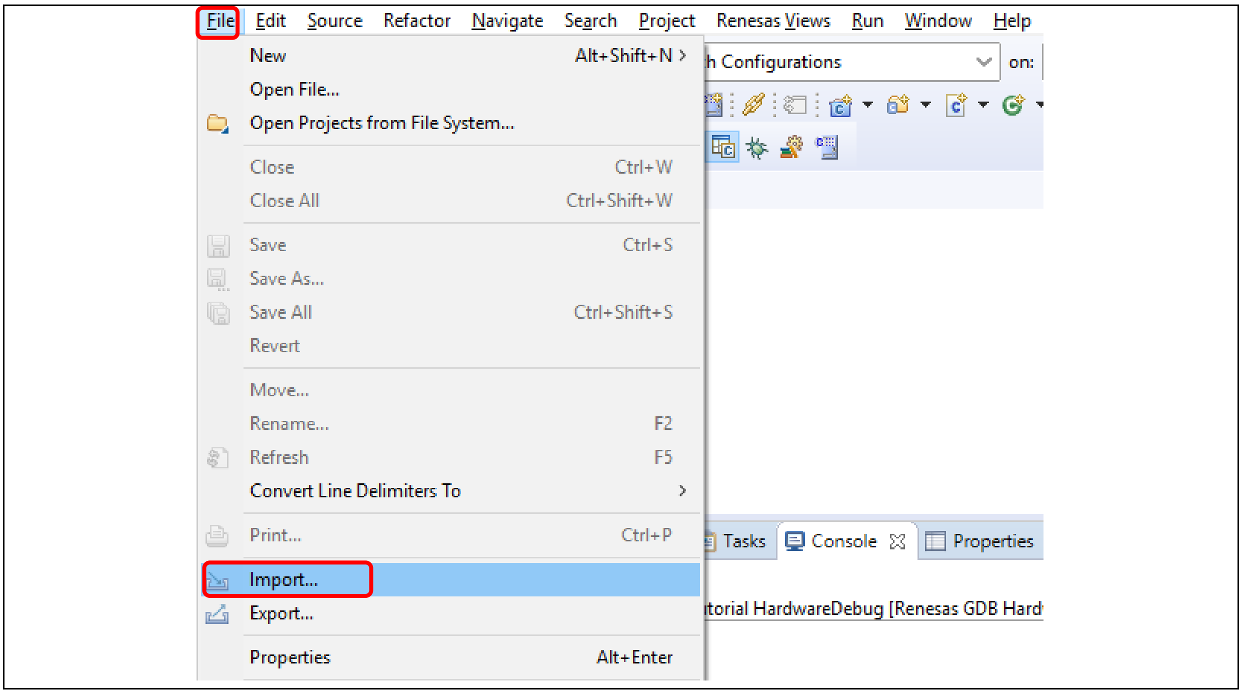
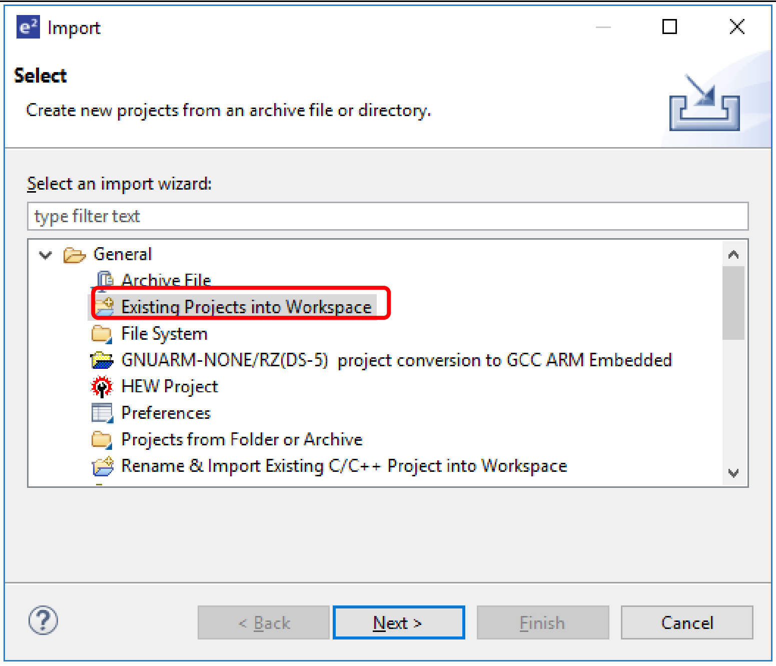


Figure 5-15 Import the sample project

(4) In the [Import] dialog, select [General] → [Existing Projects into Workspace]. Click [Next]



Import

Select

Create new projects from an archive file or directory.

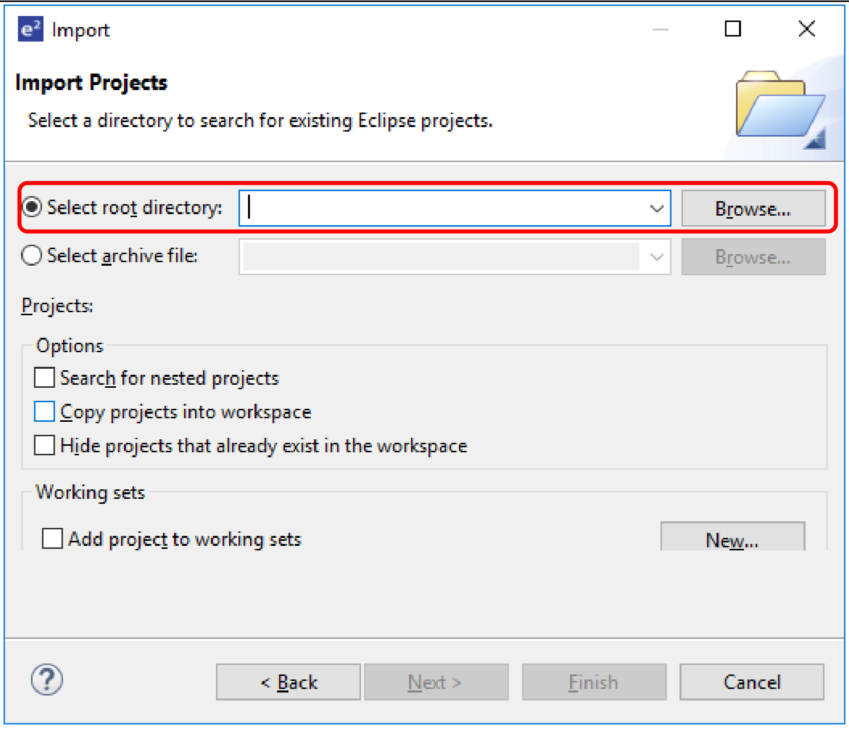
Select an import wizard:

type filter text

- General
  - Archive File
  - Existing Projects into Workspace
  - File System
  - GNUARM-NONE/RZ(DS-5) project conversion to GCC ARM Embedded
  - HEW Project
  - Preferences
  - Projects from Folder or Archive
  - Rename & Import Existing C/C++ Project into Workspace

Figure 5-16 Select import wizard

(5) In the [Import Projects] dialog, select “Select root directory”. Click [Browse] then select the folder “Tutorial” in the sample code package.



Import

Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory:

○ Select archive file:

Projects:

Options

□ Search for nested projects

□ Copy projects into workspace

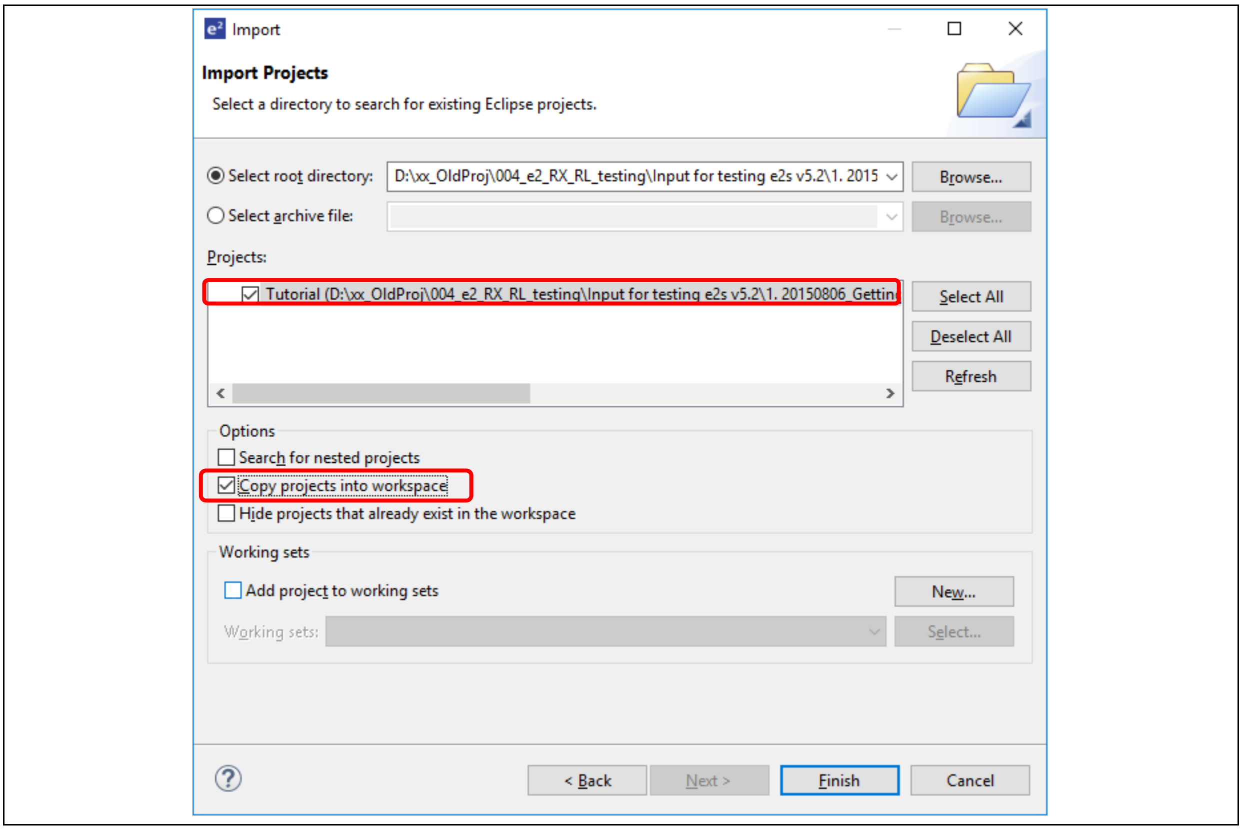
□ Hide projects that already exist in the workspace

Working sets

□ Add project to working sets

Figure 5-17 Select project location to import

(6) The project “Tutorial” will be listed in “Projects”. Check “Copy projects into workspace” then click [Finish]



Import

Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory: D:\xx_OldProj\004_e2_RX_RL_testing\Input for testing e2s v5.2\1. 2015

○ Select archive file:

Projects:

☑ Tutorial (D:\xx_OldProj\004_e2_RX_RL_testing\Input for testing e2s v5.2\1. 20150806_Gettin

Options

☐ Search for nested projects

☑ Copy projects into workspace

☐ Hide projects that already exist in the workspace

Working sets

☐ Add project to working sets

Working sets:

Figure 5-18 Complete project import

(7) Right click on the imported project and select “Upgrade Legacy e2 studio Projects…”

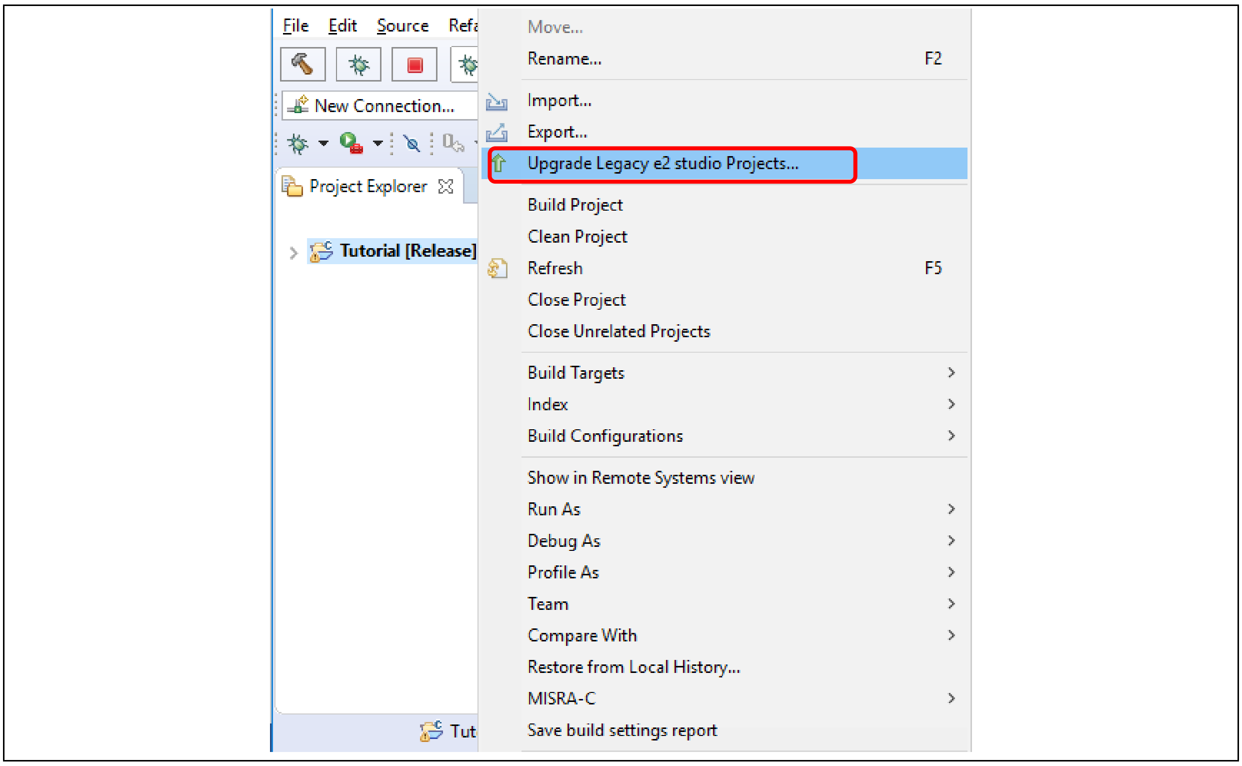
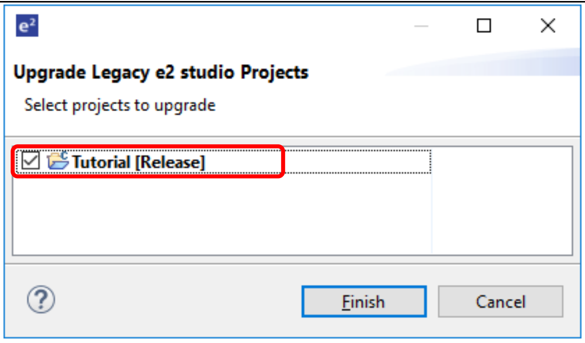


Figure 5-19 Upgrade the imported project

(8) Select “Tutorial” project and click [Finish]



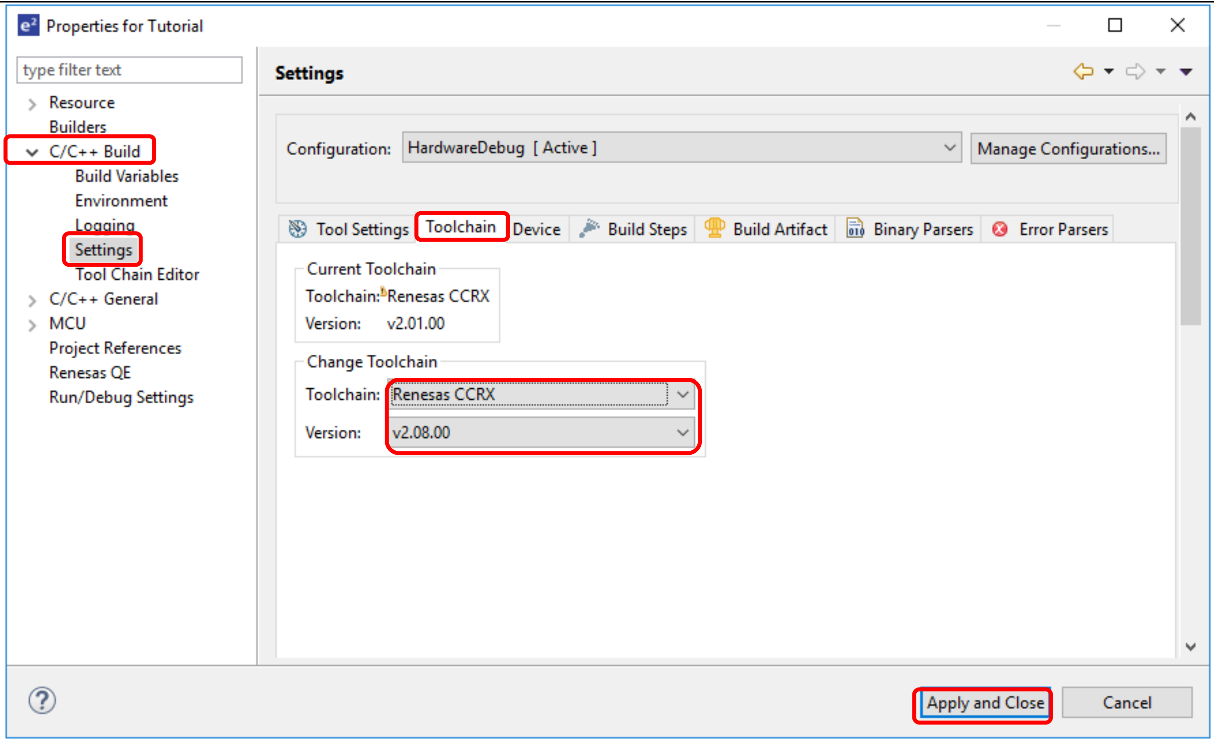
Upgrade Legacy e2 studio Projects

Select projects to upgrade

☑ Tutorial [Release]

Figure 5-20 Finish the upgrading

(9) Open the project properties, select [C/C++ Build] → [Settings] in the left pane. Select tab [Toolchain] and select the latest toolchain for the project. Click [Apply and Close].



Properties for Tutorial

Settings

Configuration: HardwareDebug [Active]

Current Toolchain
Toolchain: Renesas CCRX
Version: v2.01.00

Change Toolchain
Toolchain: Renesas CCRX
Version: v2.08.00

Figure 5-21 Update project toolchain

(10) Build the project and make sure that it is successful.

5.4.1. Breakpoints View

The Breakpoints view stores the breakpoints that were set on executable lines of a program. If a breakpoint is enabled during debugging, the execution suspends before that line of code executes. e^2 studio allows software and hardware breakpoints to be set explicitly in the IDE. Any breakpoints added via double click on the marker bar are by default hardware breakpoints. If the hardware resources are not there then the breakpoint setting will fail. In case of a hardware breakpoint setting failure, an error message will prompt the user to switch to a software breakpoint.

To select a default Hardware or Software breakpoint type:

(1) Right-click on the marker bar to pop up the context menu. For a hardware breakpoint, select [Breakpoint Types] → [e^2 studio Breakpoint]. For a software breakpoint, select [Breakpoint Types] → [C/C++ Breakpoints].

To set a breakpoint:

(1) Open "r_cg_main.c", double-click on the marker bar located in the left margin of the [C/C++ Editor] pane to set a breakpoint. A dot (Hardware breakpoint) or (Software breakpoint) is displayed in the marker bar depending on the [Breakpoint Type] selected. [Breakpoint Type] is hardware breakpoint by default.

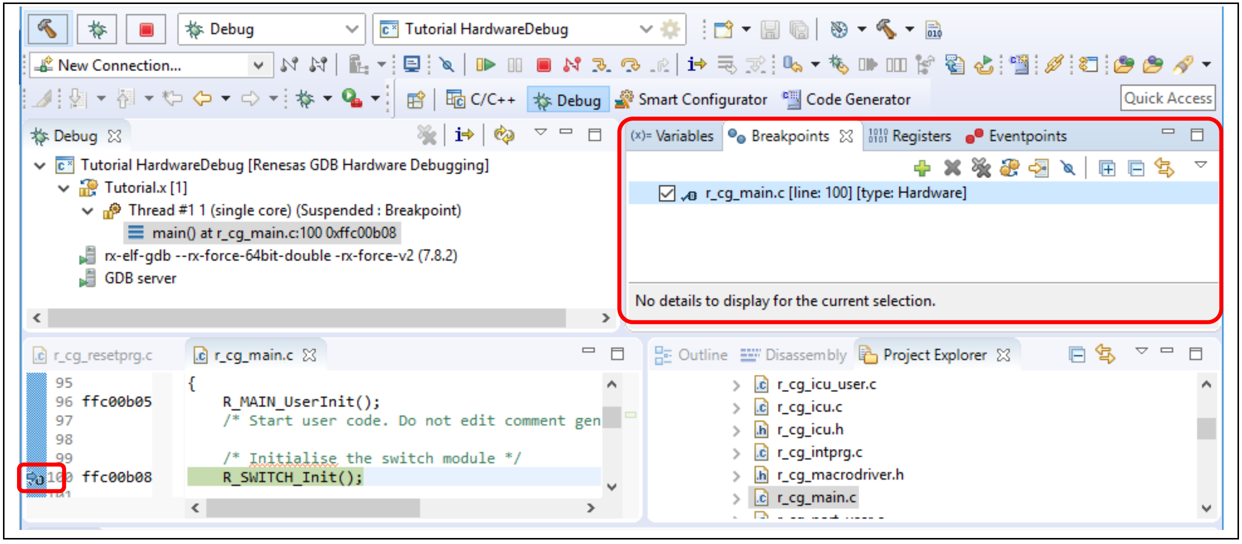
(2) Alternatively, right-click at the marker bar to choose [Toggle Hardware Breakpoint] or [Toggle Software Breakpoint] to set a hardware breakpoint or a software breakpoint .

(3) Click [Windows] → [Show View] → [Breakpoints] or icon (or use shortcut key [ALT] + [Shift] + [Q], [B]) to open the [Breakpoints] view to view the corresponding software breakpoints set. Software breakpoints can be enabled and disabled in the [Breakpoints] view.

To disable breakpoints, users can choose to disable specific breakpoints or to skip all breakpoints:

(1) To disable a specific breakpoint, right-click on the Software breakpoint or Hardware breakpoint located in the left margin of the [C/C++ Editor] pane and select [Disable Breakpoint], or uncheck the related line in the Breakpoints view. A disabled breakpoint is displayed as a white dot (or).

(2) To skip all breakpoints, click on the icon in the Breakpoints view. A blue dot with a backslash will appear in the editor pane as well as in the Breakpoints view.



☑ r_cg_main.c [line: 100] [type: Hardware]

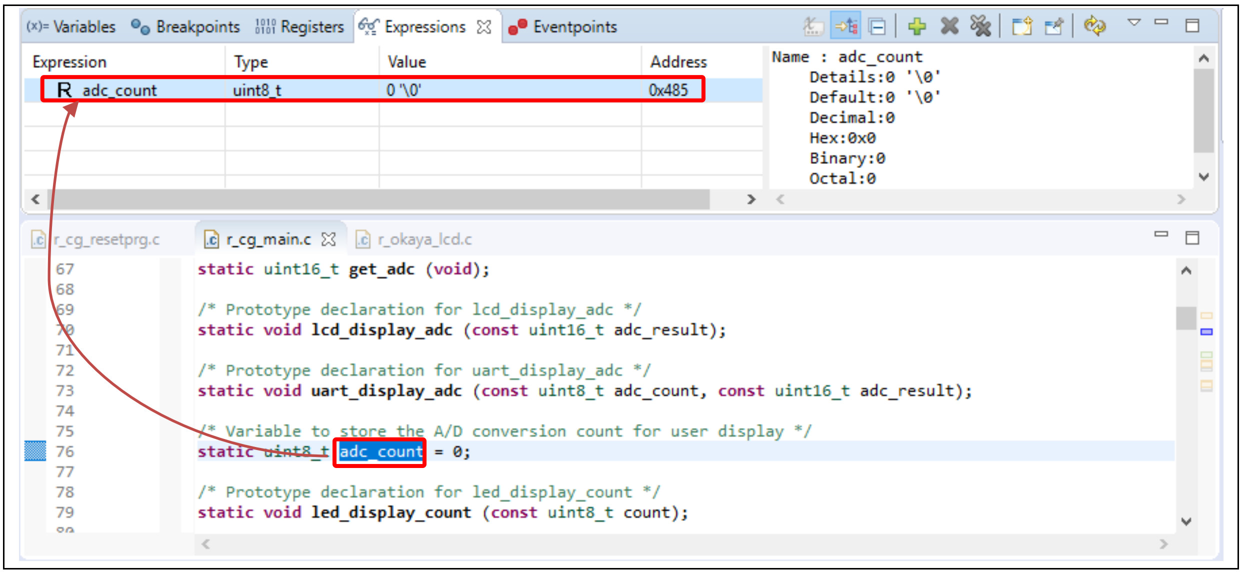
No details to display for the current selection.

```
95              {
96  ffc00b05        R_MAIN_UserInit();
97                  /* Start user code. Do not edit comment gen
98
99                  /* Initialise the switch module */
100 ffc00b08        R_SWITCH_Init();
```

Figure 5-22 [Breakpoints] view

5.4.2. Expressions View

Expressions view monitors the value of global variable, static variable or local variable during debugging. For all RX debuggers, these variables (including the local variables in scope) can be set for real-time refresh.



Expression	Type	Value	Address
R adc_count	uint8_t	0 '\0'	0x485

```
Name : adc_count
    Details:0 '\0'
    Default:0 '\0'
    Decimal:0
    Hex:0x0
    Binary:0
    Octal:0
```

```
static uint16_t get_adc (void);

/* Prototype declaration for lcd_display_adc */
static void lcd_display_adc (const uint16_t adc_result);

/* Prototype declaration for uart_display_adc */
static void uart_display_adc (const uint8_t adc_count, const uint16_t adc_result);

/* Variable to store the A/D conversion count for user display */
static uint8_t adc_count = 0;

/* Prototype declaration for led_display_count */
static void led_display_count (const uint8_t count);
```

Figure 5-23 [Expressions] View

To watch a global variable,

(1) Click [Window] → [Show View] → [Expressions] or icon 👓 to open the [Expressions] view

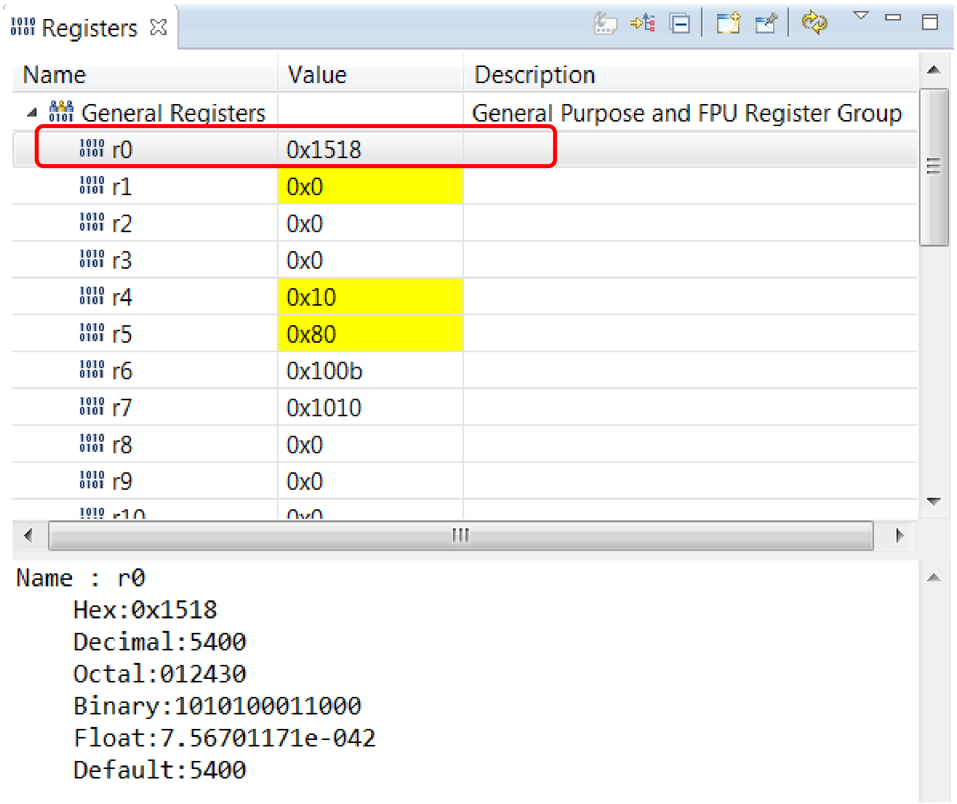
(2) Drag and drop a global variable over to the [Expressions] view. (Alternatively, right-click at the global variable to select "Add Watch Expression…"menu item to add it to the [Expressions] view).

(3) In the [Expressions] view, right-click to select "Real-time Refresh" menu item. This refresh the expression value in real-time when program is running. The character "R" indicates that this global variable will be updated in real-time.

(4) To disable the "Real-time Refresh", simply right-click to select "Disable Real-time Refresh" menu item.

5.4.3. Registers View

Registers view lists the information about the general registers of the target device. Changed values are highlighted when the program stops.



Registers

Name	Value	Description
General Registers		General Purpose and FPU Register Group
r0	0x1518	
r1	0x0	
r2	0x0	
r3	0x0	
r4	0x10	
r5	0x80	
r6	0x100b	
r7	0x1010	
r8	0x0	
r9	0x0	
r10	0x0	

```
Name : r0
    Hex:0x1518
    Decimal:5400
    Octal:012430
    Binary:1010100011000
    Float:7.56701171e-042
    Default:5400
```

Figure 5-24 [Registers] View

To view the general register “r0”,

(1) Click [Window] → [Show View] → [Registers] or icon 1010 0101 to open the [Registers] view.

(2) Click “r0” to view the values in different radix format.

Values that have been changed are highlighted (e.g. in yellow) in the [Registers] view when the program stops.

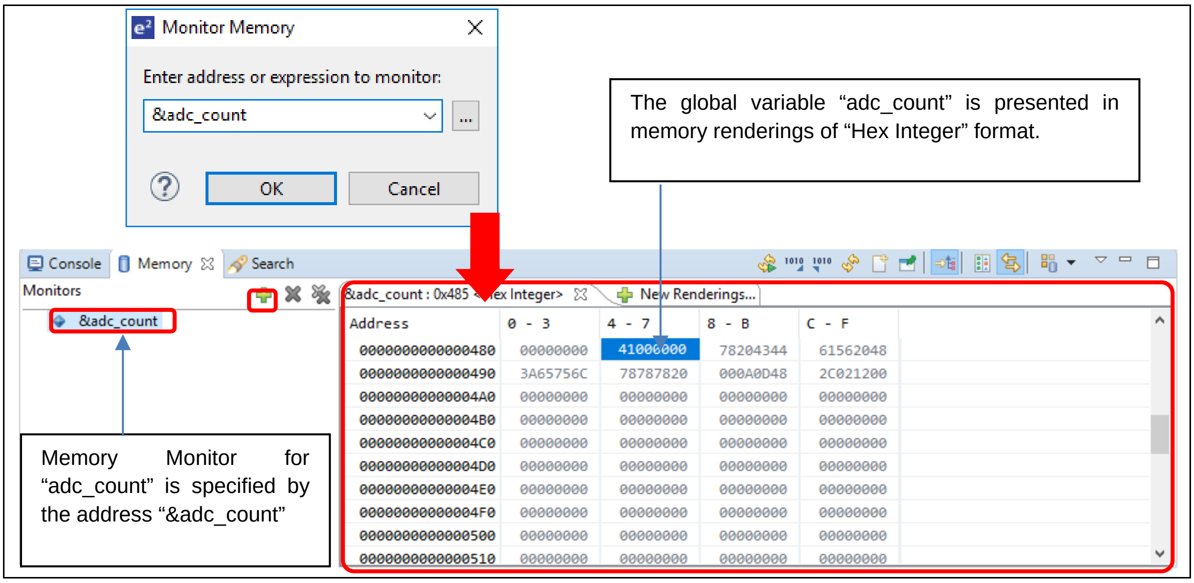
5.4.4. Memory View

Memory view allows users to view and edit the memory presented in “memory monitors”. Each monitor represents a section of memory specified by its location called “base address”. The memory data in each memory monitor can be presented in different “memory renderings”, which are the predefined data formats (e.g. Hex integer, signed integer, unsigned integer, ASCII, image etc.).

To view memory of a variable (e.g. “adc_count”),

(1) Click [Window] → [Show View] → [Memory] or icon ▯ to open the [Memory] view.

(2) Click the icon ✚ to open [Monitor Memory] dialog box. Enter the address of the variable “adc_count”.


Monitor Memory
Enter address or expression to monitor:
OK
Cancel
Console
Memory
Search
Monitors
&adc_count : 0x485 <ex Integer>
New Renderings...

Enter address or expression to monitor: &adc_count

The global variable “adc_count” is presented in memory renderings of “Hex Integer” format.

Memory Monitor for “adc_count” is specified by the address “&adc_count”

Monitors: &adc_count

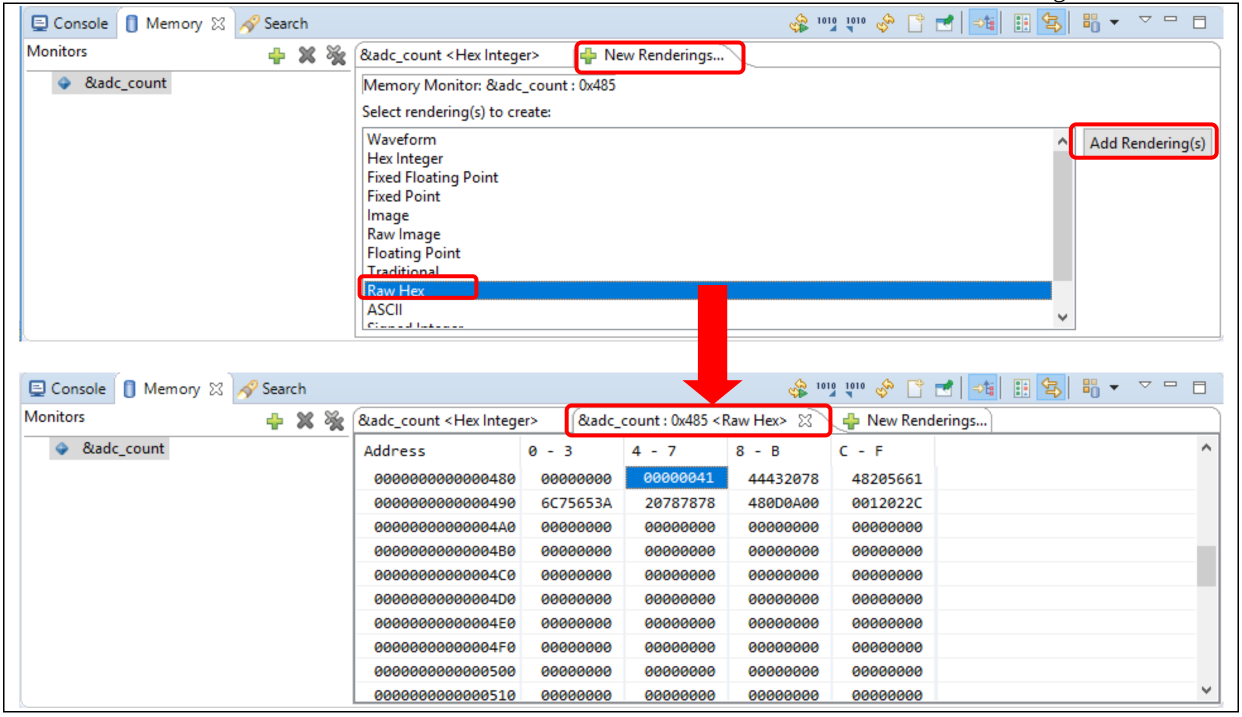
```
Address           0 - 3     4 - 7     8 - B     C - F
0000000000000480  00000000  41000000  78204344  61562048
0000000000000490  3A65756C  78787820  000A0D48  2C021200
00000000000004A0  00000000  00000000  00000000  00000000
00000000000004B0  00000000  00000000  00000000  00000000
00000000000004C0  00000000  00000000  00000000  00000000
00000000000004D0  00000000  00000000  00000000  00000000
00000000000004E0  00000000  00000000  00000000  00000000
00000000000004F0  00000000  00000000  00000000  00000000
0000000000000500  00000000  00000000  00000000  00000000
0000000000000510  00000000  00000000  00000000  00000000
```

(3) Figure 5-25 [Memory] View (1/2)

To add new renderings format (e.g. Raw Hex) for the variable “adc_count”,

(1) Click the tab New Renderings… to select “Raw Hex” to add the rendering

This creates a new tab named “&adc_count < Raw Hex>” next to the tab “&adc_count<Hex Integer>”.

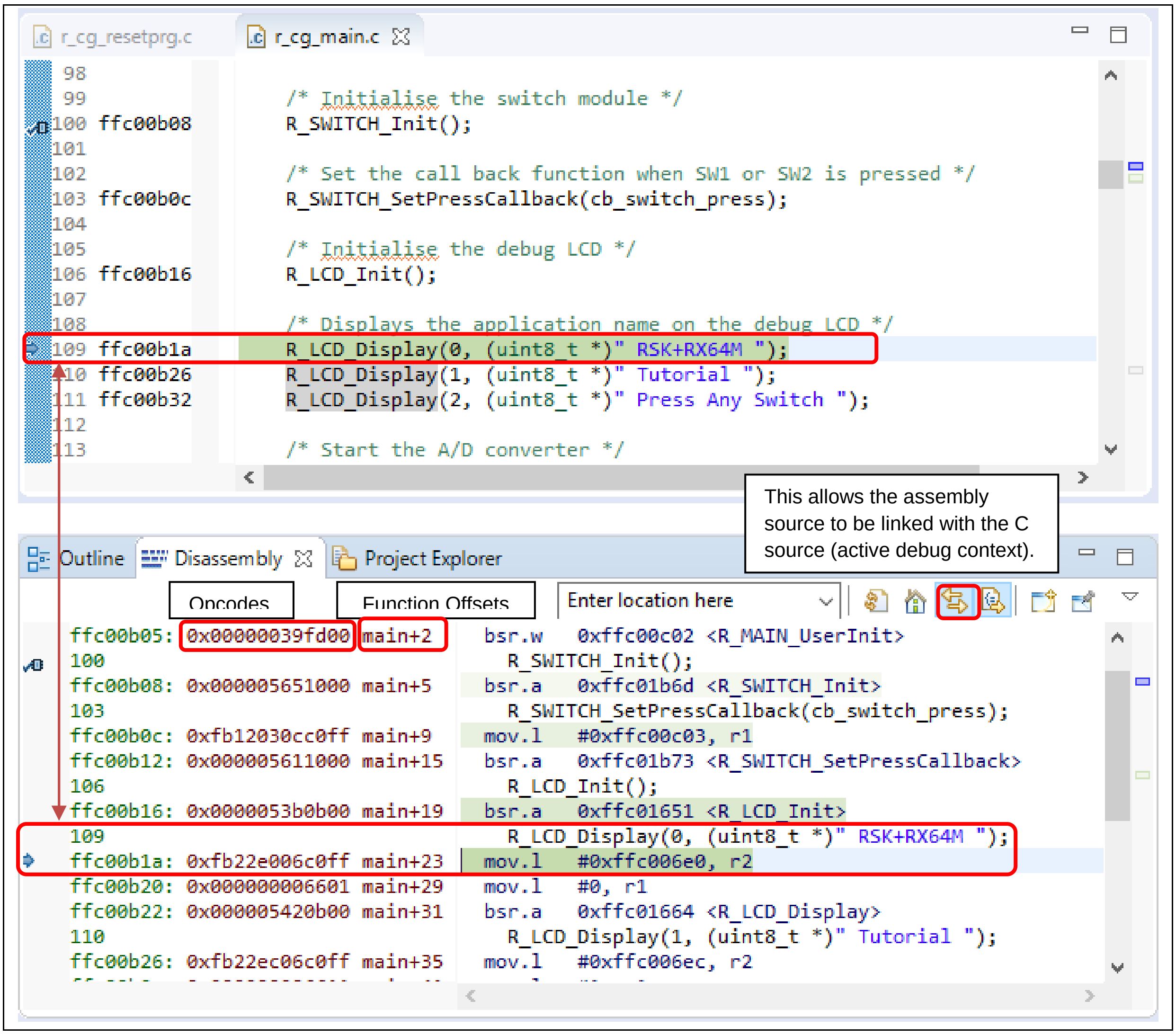


Address	0 - 3	4 - 7	8 - B	C - F
0000000000000480	00000000	00000041	44432078	48205661
0000000000000490	6C75653A	20787878	480D0A00	0012022C
00000000000004A0	00000000	00000000	00000000	00000000
00000000000004B0	00000000	00000000	00000000	00000000
00000000000004C0	00000000	00000000	00000000	00000000
00000000000004D0	00000000	00000000	00000000	00000000
00000000000004E0	00000000	00000000	00000000	00000000
00000000000004F0	00000000	00000000	00000000	00000000
0000000000000500	00000000	00000000	00000000	00000000
0000000000000510	00000000	00000000	00000000	00000000

Figure 5-26 [Memory] View (2/2)

5.4.5. Disassembly View

Disassembly view shows the loaded program as assembler instructions mixed with the source code for the comparison. Current executing line is highlighted by an arrow marker in the view. In the [Disassembly] view, user can set breakpoints at the assembler instruction, enable or disable these breakpoints, step through the disassembly instructions and even jump to a specific instruction in the program.



```
 98
 99           /* Initialise the switch module */
100 ffc00b08  R_SWITCH_Init();
101
102           /* Set the call back function when SW1 or SW2 is pressed */
103 ffc00b0c  R_SWITCH_SetPressCallback(cb_switch_press);
104
105           /* Initialise the debug LCD */
106 ffc00b16  R_LCD_Init();
107
108           /* Displays the application name on the debug LCD */
109 ffc00b1a  R_LCD_Display(0, (uint8_t *)" RSK+RX64M ");
110 ffc00b26  R_LCD_Display(1, (uint8_t *)" Tutorial ");
111 ffc00b32  R_LCD_Display(2, (uint8_t *)" Press Any Switch ");
112
113           /* Start the A/D converter */
```

This allows the assembly source to be linked with the C source (active debug context).

```
ffc00b05: 0x00000039fd00 main+2    bsr.w   0xffc00c02 <R_MAIN_UserInit>
100                                  R_SWITCH_Init();
ffc00b08: 0x000005651000 main+5    bsr.a   0xffc01b6d <R_SWITCH_Init>
103                                  R_SWITCH_SetPressCallback(cb_switch_press);
ffc00b0c: 0xfb12030cc0ff main+9    mov.l   #0xffc00c03, r1
ffc00b12: 0x000005611000 main+15   bsr.a   0xffc01b73 <R_SWITCH_SetPressCallback>
106                                  R_LCD_Init();
ffc00b16: 0x0000053b0b00 main+19   bsr.a   0xffc01651 <R_LCD_Init>
109                                  R_LCD_Display(0, (uint8_t *)" RSK+RX64M ");
ffc00b1a: 0xfb22e006c0ff main+23   mov.l   #0xffc006e0, r2
ffc00b20: 0x000000006601 main+29   mov.l   #0, r1
ffc00b22: 0x000005420b00 main+31   bsr.a   0xffc01664 <R_LCD_Display>
110                                  R_LCD_Display(1, (uint8_t *)" Tutorial ");
ffc00b26: 0xfb22ec06c0ff main+35   mov.l   #0xffc006ec, r2
```

Figure 5-27 [Disassembly] View

To view both C and assembly codes in a mixed mode,

(1) Click [Window] → [Show View] → [Disassembly] or icon ☰ to open the [Disassembly] view

(2) Click icon ⇆ to enable the synchronization between assembly source and the C source (active debug context).

(3) In [Disassembly] view, right-click at the address column to select “Show Opcodes” and “Show Function Offsets”.

(4) You can enable source addresses within the editor using the context menu.

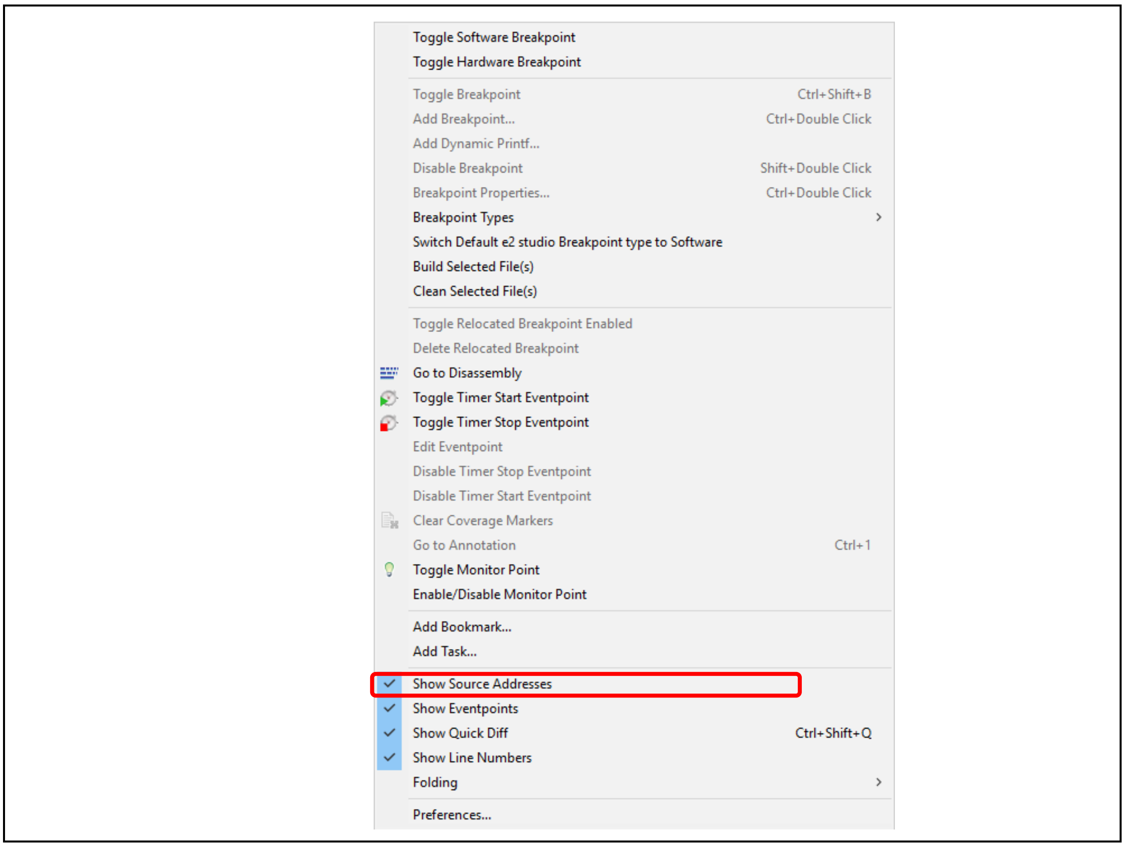
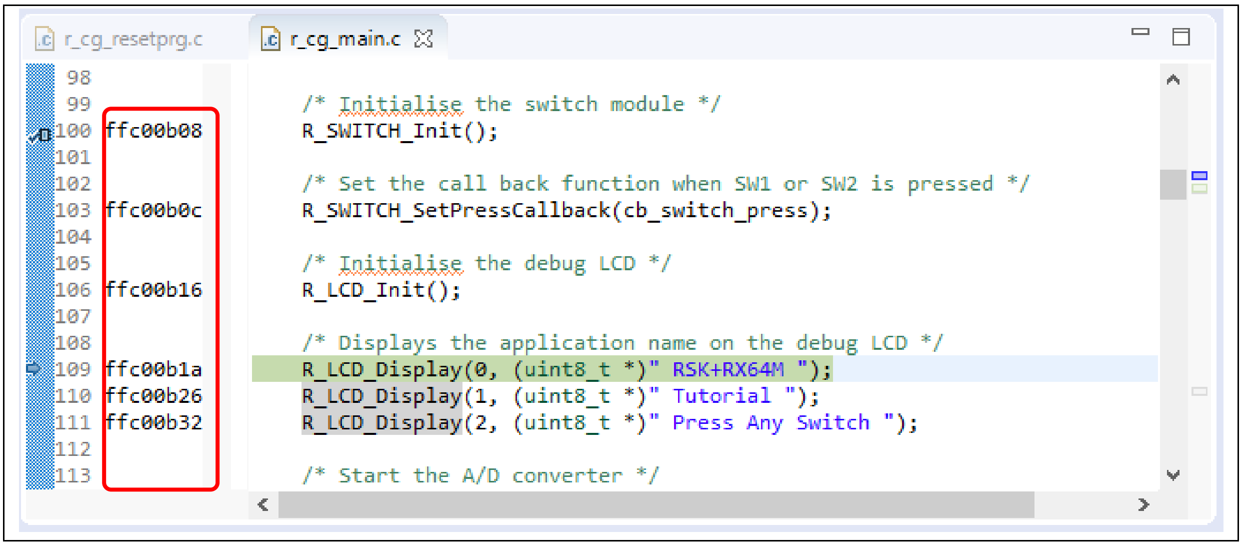

Toggle Software Breakpoint
Toggle Hardware Breakpoint
Toggle Breakpoint Ctrl+Shift+B
Add Breakpoint... Ctrl+Double Click
Add Dynamic Printf...
Disable Breakpoint Shift+Double Click
Breakpoint Properties... Ctrl+Double Click
Breakpoint Types
Switch Default e2 studio Breakpoint type to Software
Build Selected File(s)
Clean Selected File(s)
Toggle Relocated Breakpoint Enabled
Delete Relocated Breakpoint
Go to Disassembly
Toggle Timer Start Eventpoint
Toggle Timer Stop Eventpoint
Edit Eventpoint
Disable Timer Stop Eventpoint
Disable Timer Start Eventpoint
Clear Coverage Markers
Go to Annotation Ctrl+1
Toggle Monitor Point
Enable/Disable Monitor Point
Add Bookmark...
Add Task...
✓ Show Source Addresses
✓ Show Eventpoints
✓ Show Quick Diff Ctrl+Shift+Q
✓ Show Line Numbers
Folding
Preferences...

Figure 5-28 Source Addresses Menu



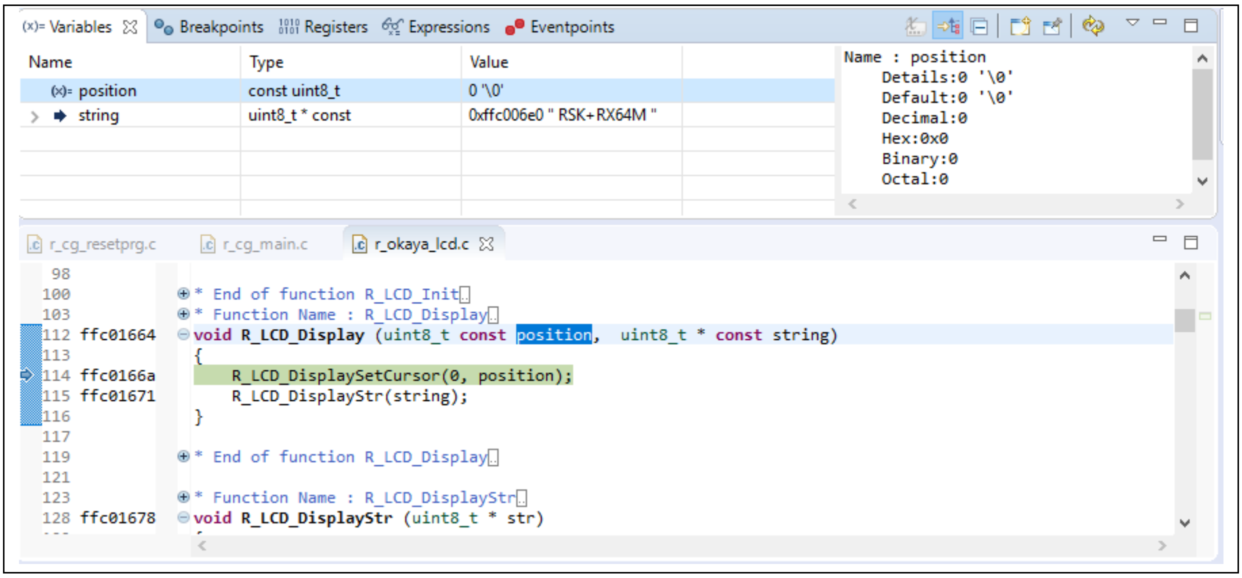
```
                /* Initialise the switch module */
ffc00b08        R_SWITCH_Init();

                /* Set the call back function when SW1 or SW2 is pressed */
ffc00b0c        R_SWITCH_SetPressCallback(cb_switch_press);

                /* Initialise the debug LCD */
ffc00b16        R_LCD_Init();

                /* Displays the application name on the debug LCD */
ffc00b1a        R_LCD_Display(0, (uint8_t *)" RSK+RX64M ");
ffc00b26        R_LCD_Display(1, (uint8_t *)" Tutorial ");
ffc00b32        R_LCD_Display(2, (uint8_t *)" Press Any Switch ");

                /* Start the A/D converter */
```

Figure 5-29 Source Addresses displayed in Editor

5.4.6. Variables View

Variables view displays all the valid local variables in the current program scope.

Please refer to ‘Expressions’ view to watch global variables or external variables out of current program scope.



Name	Type	Value
position	const uint8_t	0 '\0'
string	uint8_t * const	0xffc006e0 " RSK+RX64M "

```
Name : position
    Details:0 '\0'
    Default:0 '\0'
    Decimal:0
    Hex:0x0
    Binary:0
    Octal:0
```

```
         * End of function R_LCD_Init
         * Function Name : R_LCD_Display
ffc01664 void R_LCD_Display (uint8_t const position,  uint8_t * const string)
         {
ffc0166a     R_LCD_DisplaySetCursor(0, position);
ffc01671     R_LCD_DisplayStr(string);
         }

         * End of function R_LCD_Display

         * Function Name : R_LCD_DisplayStr
ffc01678 void R_LCD_DisplayStr (uint8_t * str)
```

Figure 5-30 [Variables] View

To observe a local variable (e.g. “position” for function “R_LCD_Display()”),

(1) Click [Window] → [Show View] → [Variables] or icon (x)= to open the [Variables] view.

(2) Step into the function “R_LCD_Display ()” to view the value of local variable “position”.

Note:
The variables which optimized out or temporary allocated to accumulator registers may not appear in this view. Please refer to Disassembly view if necessary.

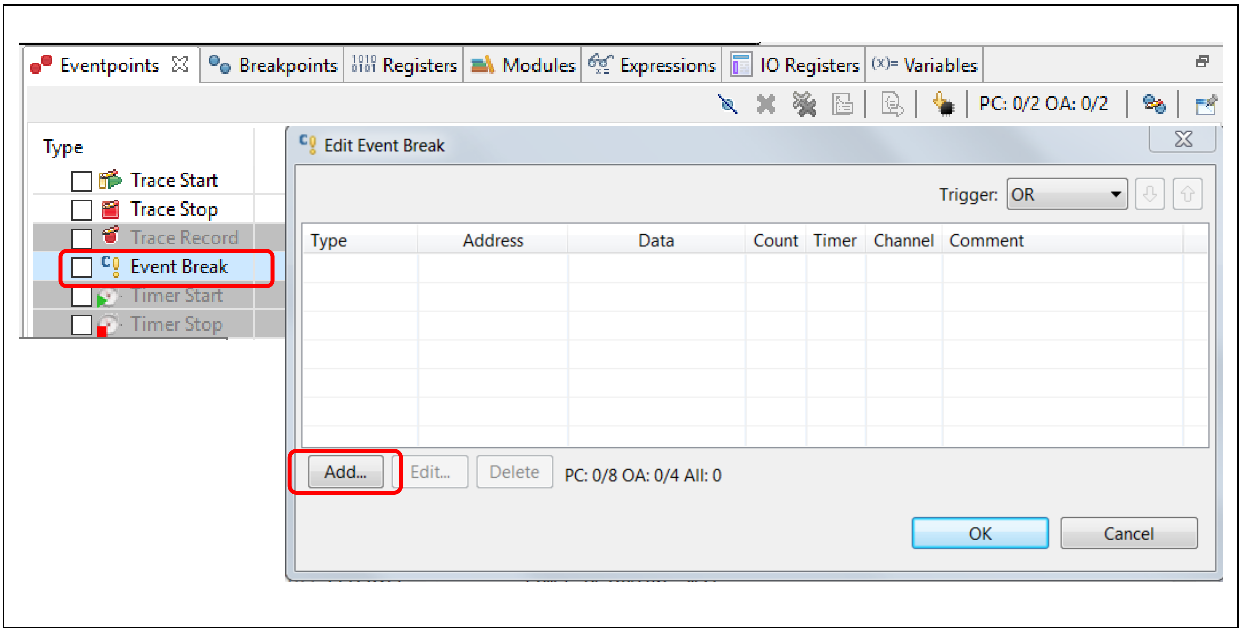
5.4.7. Eventpoints View

An event refers to a combination of conditions set for executing break or trace features during program execution. [Eventpoints] view enables user to set up or view defined events of different category e.g. trace start, trace stop, trace record, event break, before PC, performance (timer) start and performance (timer) stop.

The number of events that can be set and the setting conditions differ with each MCU. These are two (2) types of events:

- Execution address: The emulator detects execution of the instruction at the specified address by the CPU. It can be a "before PC" break (e.g. with event condition is satisfied immediately before execution of the instruction at the specified address) or other events (e.g. with event condition is satisfied immediately after execution of the instruction at the specified address).
- Data access: The emulator detects access under a specified condition to specified address or specified address range. This allows to setup complex address and data matching criteria.

Event combination (e.g. OR, AND (cumulative) and Sequential) can be applied to two (2) or more events.



Trigger: OR

PC: 0/8 OA: 0/4 All: 0

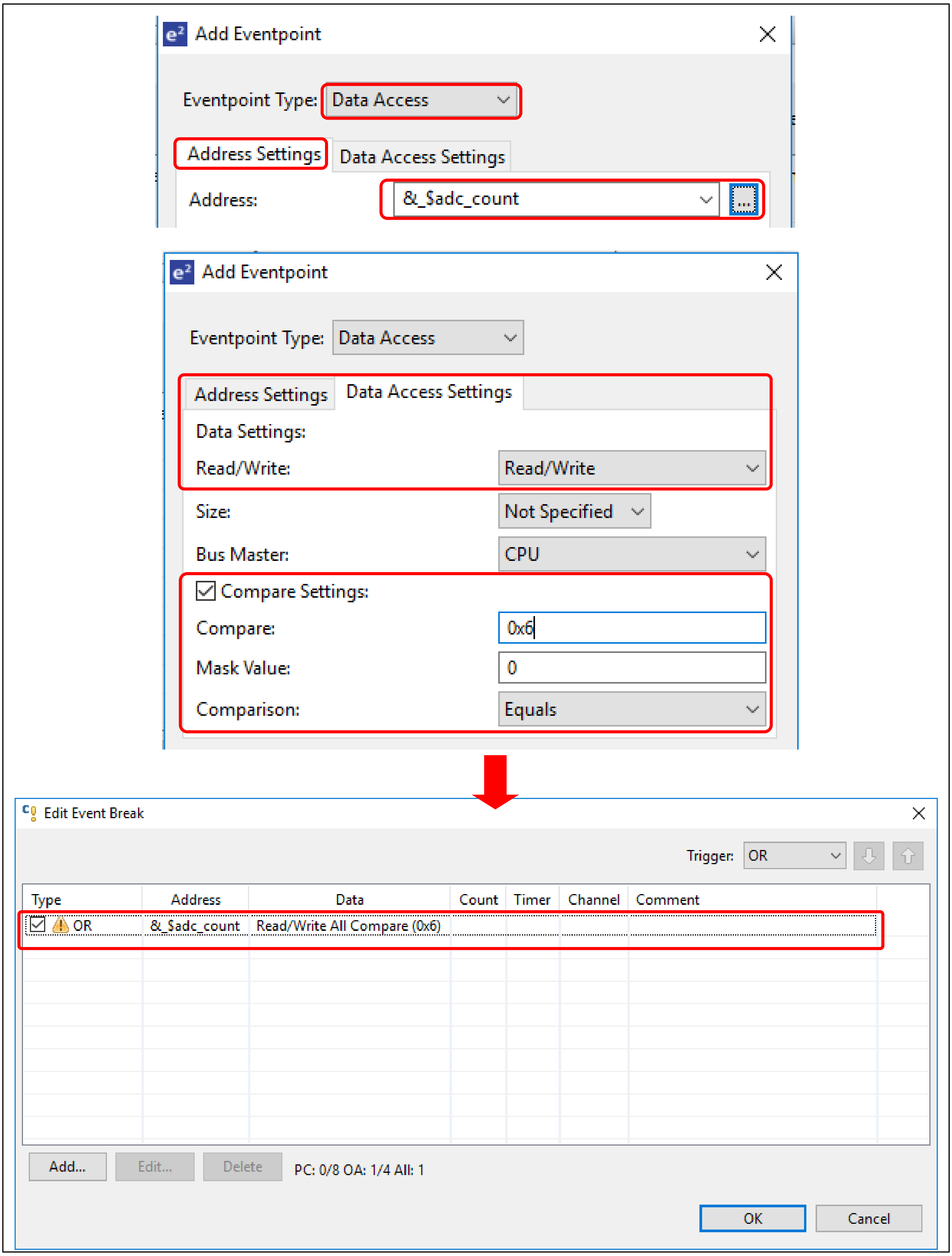
Figure 5-31 [Eventpoints] View (1/2)

To set an event break for a global variable when address/data is matched (e.g. when adc_count = "0x6"),

(1) Click [Window] → [Show View] → [Eventpoints] or icon ● to open the [Eventpoints] view.

(2) Double-click at "Event Break" option to open [Edit Event Break] dialog box

(3) Click [Add…] button to continue.



Add Eventpoint

Eventpoint Type: Data Access

Address Settings | Data Access Settings

Address: &_$adc_count

Add Eventpoint

Eventpoint Type: Data Access

Address Settings | Data Access Settings

Data Settings:

Read/Write: Read/Write

Size: Not Specified

Bus Master: CPU

☑ Compare Settings:

Compare: 0x6

Mask Value: 0

Comparison: Equals

Edit Event Break

Trigger: OR

Type	Address	Data	Count	Timer	Channel	Comment
☑ OR	&_$adc_count	Read/Write All Compare (0x6)				

PC: 0/8 OA: 1/4 All: 1

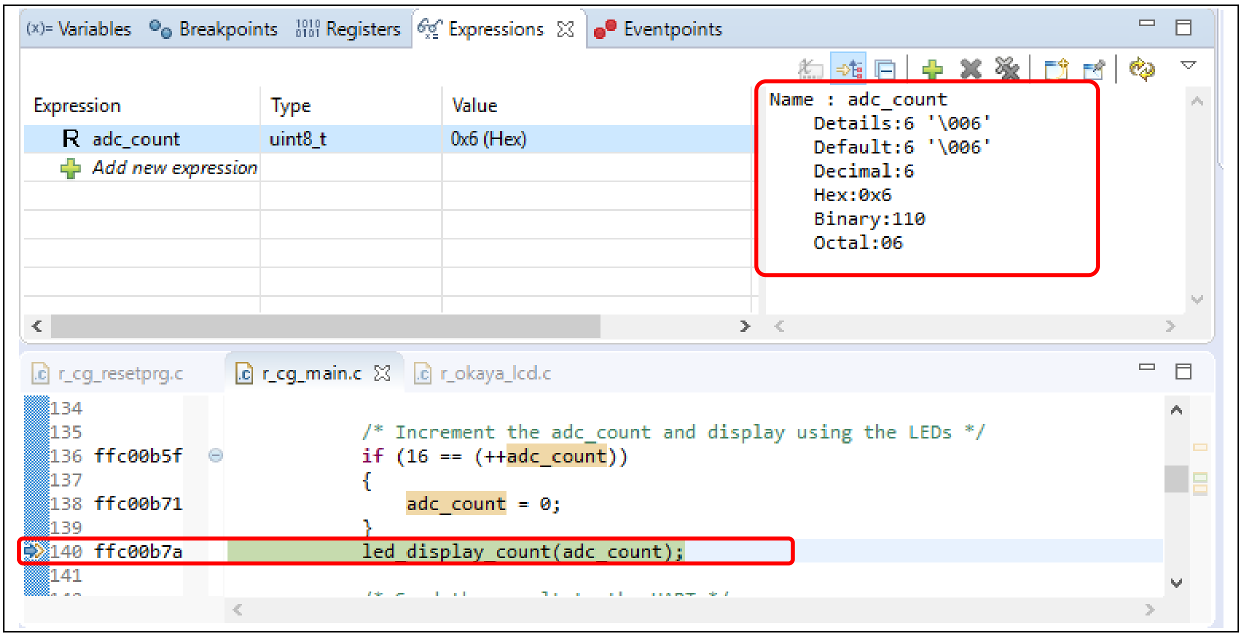
Figure 5-32 [Eventpoints] View (2/2)

(4) Select “Data Access” as the eventpoint type.

(5) Go to the [Address Settings] tab, click the icon ... to browse for the symbol “_$adc_count”. (The address of this global variable is “&_$adc_count”)

(6) Next, switch to the [Data Access Settings] tab, enable the [Compare Settings] checkbox and set the compare value equals to “0x6”. Click [OK] to proceed.

(7) Ensure that the event break for “adc_count = 0x6” is set and enabled in the [Eventpoints] view. Reset to execute the program from the start. Press SW1 6 times.



Expression	Type	Value
R adc_count	uint8_t	0x6 (Hex)
Add new expression		

```
Name : adc_count
    Details:6 '\006'
    Default:6 '\006'
    Decimal:6
    Hex:0x6
    Binary:110
    Octal:06
```

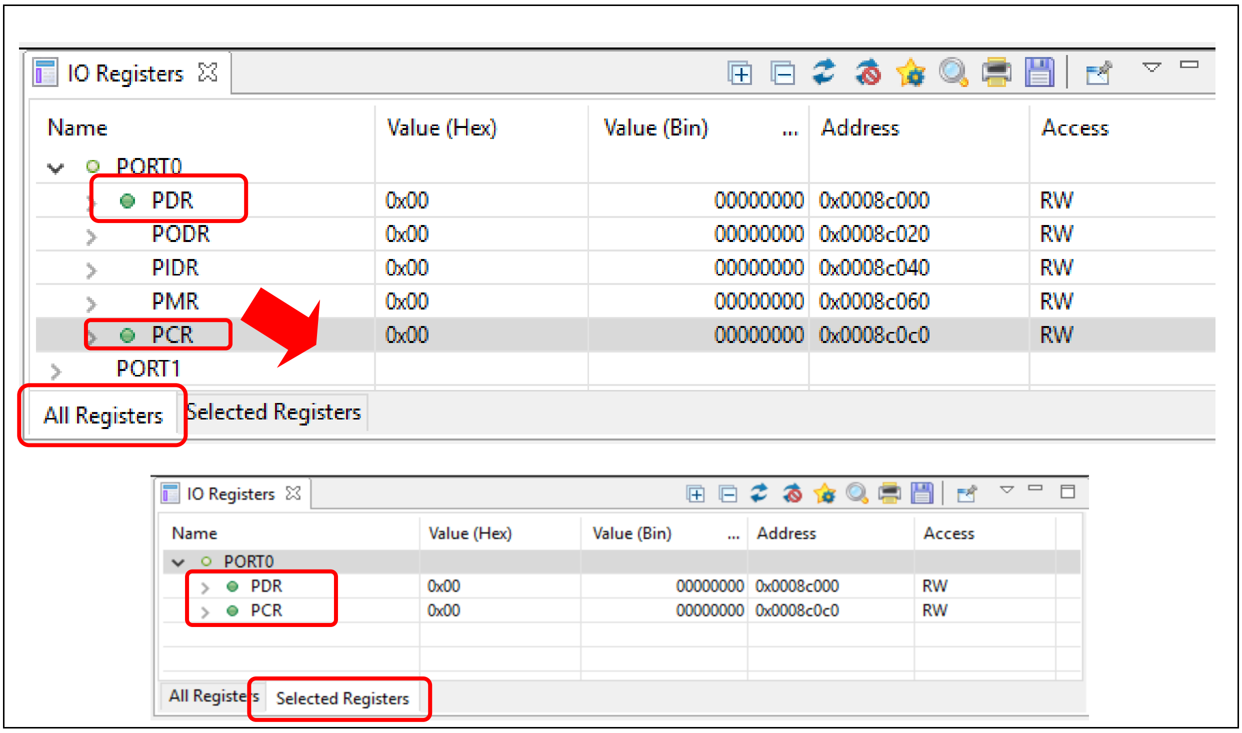
```
134
135                       /* Increment the adc_count and display using the LEDs */
136 ffc00b5f              if (16 == (++adc_count))
137                       {
138 ffc00b71                  adc_count = 0;
139                       }
140 ffc00b7a              led_display_count(adc_count);
141
```

Figure 5-33 Execution of Event Break

Figure 5-22 shows that when adc_count reaches the value of 6 (or 0x6), the program stops at code line No.140 (right after the line of code increasing adc_count).

5.4.8. IO Registers View

IO Registers is also known as the Special Function Registers (SFR). The [IO Register] view displays all the registers set defined in a target-specific IO file, including their address, hex and binary value. User can further customize own [IO registers] view by adding IO registers selectively to the [Selected Registers] pane.



Name	Value (Hex)	Value (Bin) ...	Address	Access
PORT0				
● PDR	0x00	00000000	0x0008c000	RW
PODR	0x00	00000000	0x0008c020	RW
PIDR	0x00	00000000	0x0008c040	RW
PMR	0x00	00000000	0x0008c060	RW
● PCR	0x00	00000000	0x0008c0c0	RW
PORT1				

Name	Value (Hex)	Value (Bin) ...	Address	Access
PORT0				
● PDR	0x00	00000000	0x0008c000	RW
● PCR	0x00	00000000	0x0008c0c0	RW

Figure 5-34 [IO Registers] View

To view selected IO registers (e.g. PDR and PCR in PORT0),

(1) Click [Windows] → [Show View] → [Others…]. In “Show View” dialog, click [IO Registers] under [Debug] or icon ▤ to open the [IO Registers] view

(2) Under the [All Registers] tab, locate [PORT0] in the [IO Registers] view. Expand the PORT0 IO register list.

(3) Drag and drop the “PDR” and “PCR” to the [Selected Registers] pane. A green dot ● besides the IO register indicates the status of being the selected register(s).

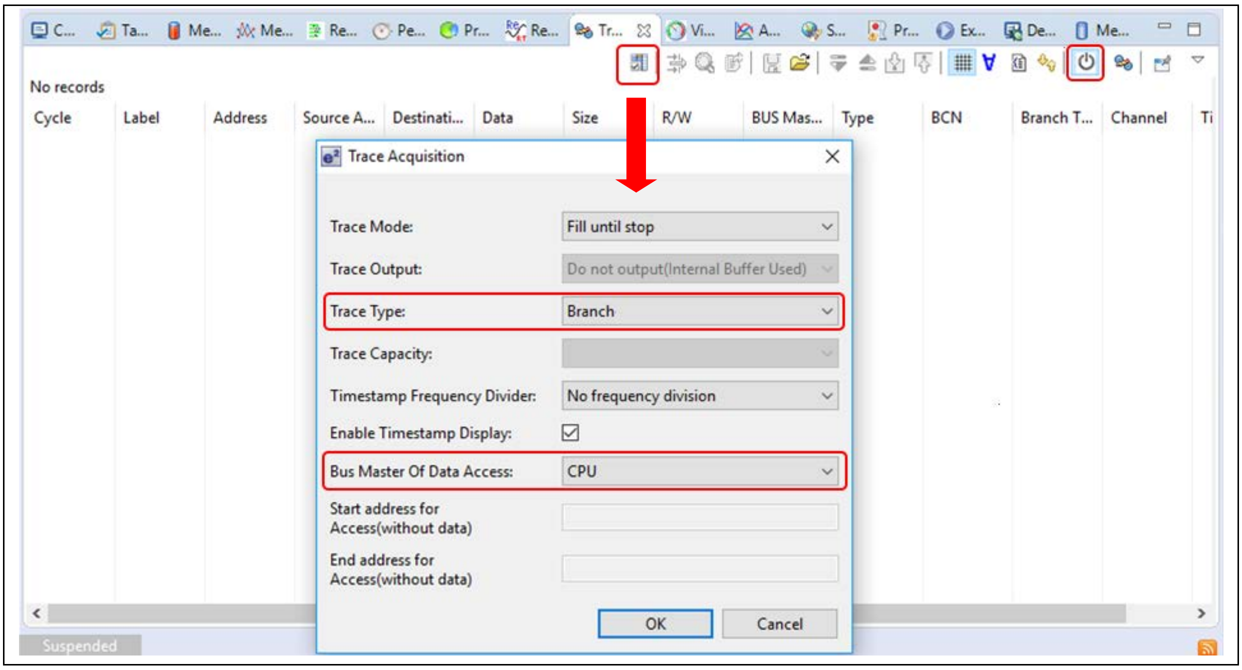
(4) Switch to the [Selected Registers] tab to view “PDR” and “PCR” of the “PORT0” IO register

The expanded IO register list may take a longer time to load in the [All Registers] pane. Hence, it is advisable to customize and view multiple selected IO registers from the [Selected Registers] pane.

5.4.9. Trace View

Tracing means the acquisition of bus information per cycle from the trace memory during user program execution. The acquired trace information is displayed in the [Trace] view. It helps user to track the program execution flow to search for and examine the points where problems arise.

The trace buffer is limited (with size of 1 to 32 Mbytes), oldest trace data is overwritten with the new data after the buffer has become full.



Trace Acquisition

Trace Mode: Fill until stop

Trace Output: Do not output(Internal Buffer Used)

Trace Type: Branch

Trace Capacity:

Timestamp Frequency Divider: No frequency division

Enable Timestamp Display: ☑

Bus Master Of Data Access: CPU

Start address for Access(without data):

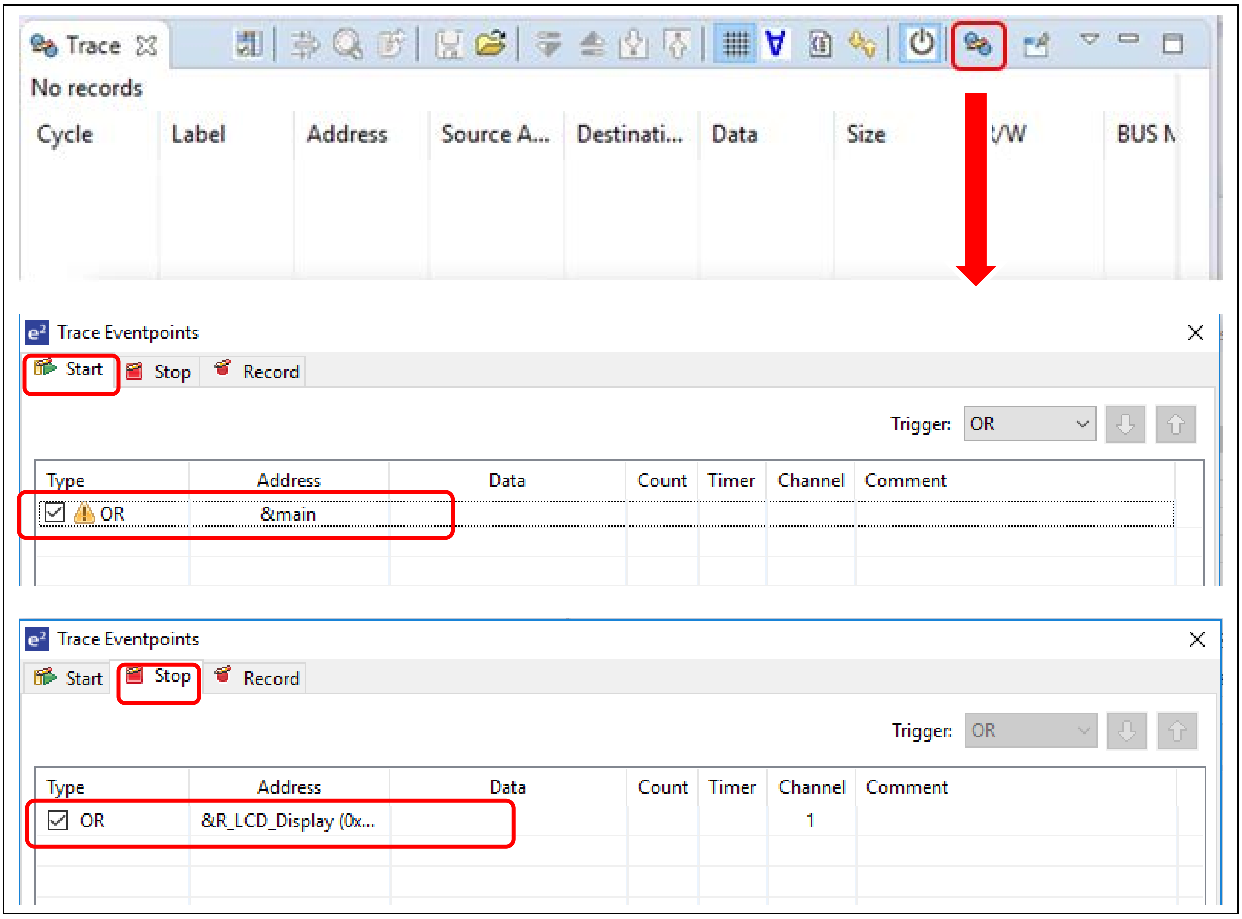
End address for Access(without data):

OK Cancel

Figure 5-35 [Trace] View (1/2)

To set a point-to-point trace between the two (2) functions (e.g. tracing from function “main()” to “R_LCD_Display()”),

- Click [Windows] → [Show View] → [Others…]. In “Show View” dialog, click [Trace] under [Debug] or icon to open the [Trace] view.
- Turn on the Trace view by selecting the ⏻ icon.
- Click icon (Acquisition) to set
  - Trace Mode: "Fill until stop”
  - Trace Type: “Branch”
  - Bus Master Of Data Access: “CPU”
- Click [OK] to proceed.



Trace

No records

Cycle	Label	Address	Source A...	Destinati...	Data	Size	:/W	BUS N

Trace Eventpoints

Start Stop Record

Trigger: OR

Type	Address	Data	Count	Timer	Channel	Comment
☑ ⚠ OR	&main					

Trace Eventpoints

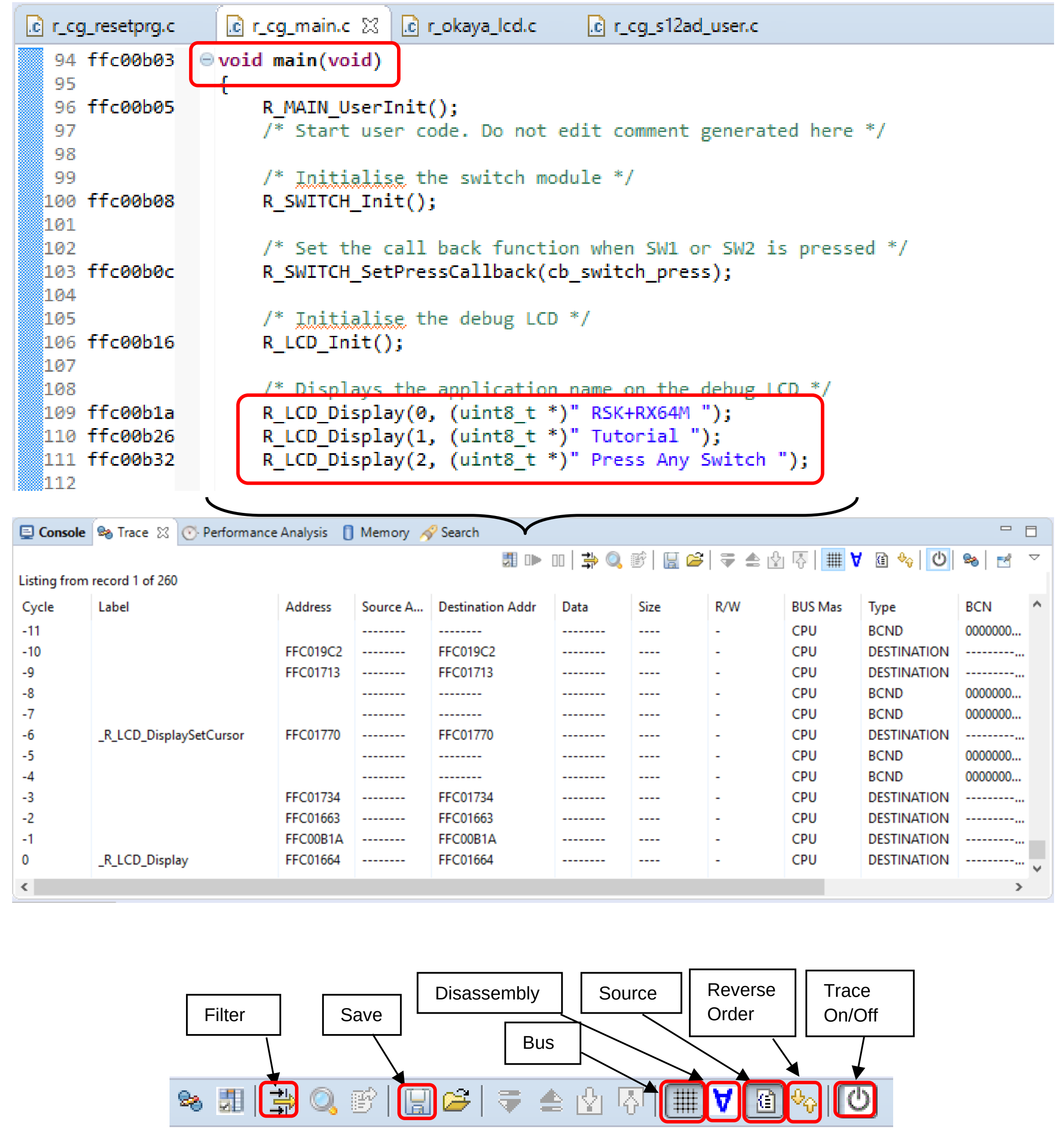
Start Stop Record

Trigger: OR

Type	Address	Data	Count	Timer	Channel	Comment
☑ OR	&R_LCD_Display (0x...				1	

Figure 5-36 [Trace] View (2/2)

- Click (Edit Trace Event Points) to open [Trace Eventpoints] dialog box
- Under the [Start] tab, add the 1st event point at “main()” function (by the execution address “&main”).
- Then, switch to [Stop] tab, add the 2nd event point at “R_LCD_Display()” function (by the execution address “&R_LCD_Display”).
- Next, execute the program after reset.



```
ffc00b03  void main(void)
          {
ffc00b05      R_MAIN_UserInit();
              /* Start user code. Do not edit comment generated here */

              /* Initialise the switch module */
ffc00b08      R_SWITCH_Init();

              /* Set the call back function when SW1 or SW2 is pressed */
ffc00b0c      R_SWITCH_SetPressCallback(cb_switch_press);

              /* Initialise the debug LCD */
ffc00b16      R_LCD_Init();

              /* Displays the application name on the debug LCD */
ffc00b1a      R_LCD_Display(0, (uint8_t *)" RSK+RX64M ");
ffc00b26      R_LCD_Display(1, (uint8_t *)" Tutorial ");
ffc00b32      R_LCD_Display(2, (uint8_t *)" Press Any Switch ");
```

Listing from record 1 of 260

Cycle	Label	Address	Source A...	Destination Addr	Data	Size	R/W	BUS Mas	Type	BCN
-11			--------	--------	--------	----	-	CPU	BCND	0000000...
-10		FFC019C2	--------	FFC019C2	--------	----	-	CPU	DESTINATION	---------...
-9		FFC01713	--------	FFC01713	--------	----	-	CPU	DESTINATION	---------...
-8			--------	--------	--------	----	-	CPU	BCND	0000000...
-7			--------	--------	--------	----	-	CPU	BCND	0000000...
-6	_R_LCD_DisplaySetCursor	FFC01770	--------	FFC01770	--------	----	-	CPU	DESTINATION	---------...
-5			--------	--------	--------	----	-	CPU	BCND	0000000...
-4			--------	--------	--------	----	-	CPU	BCND	0000000...
-3		FFC01734	--------	FFC01734	--------	----	-	CPU	DESTINATION	---------...
-2		FFC01663	--------	FFC01663	--------	----	-	CPU	DESTINATION	---------...
-1		FFC00B1A	--------	FFC00B1A	--------	----	-	CPU	DESTINATION	---------...
0	_R_LCD_Display	FFC01664	--------	FFC01664	--------	----	-	CPU	DESTINATION	---------...

Figure 5-37 Point-to-Point Trace between Two Functions

The figure above shows the trace result from function “main()” to “R_LCD_Display()”. The trace result can be filtered by the key trace parameters (e.g. branch type, address range) and saved to the .xml format (with the inclusion of bus, assembly and source information).

Note:

> External trace feature of RX device with E20 emulator works only through Mictor-38pin interface. However, it is not available through 14pin JTAG/FINE interface, even with E20 emulator. RX emulator interface specifications can be downloaded at the following site.
> https://www.renesas.com/en-sg/search/keyword-search.html#q=R20UT0399

CHAPTER 6. HELP

The help system allows user to browse, search, bookmark and print help documentation from a separate Help window or Help view within the workbench. User can also access online forum dedicated to e^2 studio from here.

Click on [Help] tap to pull down Help menu.

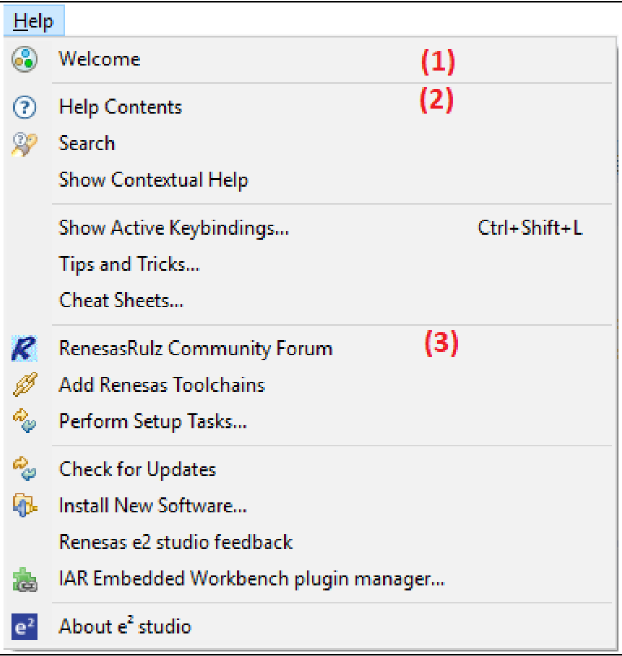


Figure 6-1 Help Menu

Quick Help Tips

① Click [Welcome] for Overview of e^2 studio, link to access IDE tutorial and sample, and to view Release Notes.

② Click [Help Contents] to open a separate Help window with search function.

③ Click [RenesasRulz Community Forum] to go online forum that is dedicated to topics and discussion related to e^2 studio IDE. Internet connection is required.

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	July 20, 2018	-	First Edition issued, Supporting e^2 studio IDE v7.0.0
		-	

e^2 studio User's Manual: Getting Started Guide (for V7.x)

Publication Date: Rev.1.00 July 20, 2018

Published by: Renesas Electronics Corporation

SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
1001 Murphy Ranch Road, Milpitas, CA 95035, U.S.A
Tel: +1-408-432-8888, Fax: +1-408-434-5351
Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004
Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900
Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327
Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679
Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P.R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999
Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022
Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670
Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300
Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510
Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HALII Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777
Renesas Electronics Korea Co., Ltd.

© 2017 Renesas Electronics Corporation. All rights reserved

Colophon 5.0

e^2 studio

Renesas Electronics Corporation