User's Manual

e² studio 2020-04 and e² studio v7.8

Integrated Development Environment

RENESAS MCU
RX, RL78, RH850 Family

All information contained in these materials, including products and product specifications, represents information on the product at the time of publication and is subject to change by Renesas Electronics Corp. without notice. Please review the latest information published by Renesas Electronics Corp. through various means, including the Renesas Electronics Corp. website (http://www.renesas.com).

Renesas Electronics
www.renesas.com

Rev.1.00 May 2020

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 or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving 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, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. 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 shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended 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; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or 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 damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, 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 and impractical, you are responsible for evaluating 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. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall 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 shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, 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.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan

www.renesas.com

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

© 2020 Renesas Electronics Corporation. All rights reserved.

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Table of Contents

1. General 1
1.1 System Configuration 1
1.2 Operating Environment 1
1.2.1 System Requirements 1
1.2.2 Supported Toolchains 1
1.3 Supported Emulator Device 2
1.4 Supported Simulator 2

2. Installation 3
2.1 Installation of e^2 studio IDE (64-bit version) 3
2.2 Installation of e2 studio IDE (32-bit version) 9
2.3 Un-installation of e^2 studio IDE 13
2.4 Update e^2 studio plugins 14
2.4.1 Online Minor Version Update 14
2.4.2 Offline Minor Version Update 17
2.5 Installation of Compiler Package 18

3. Project generation 19
3.1 New Project Generation 19
3.2 New Debug Only Project Generation (RH850) 24
3.3 Import Existing Projects into Workspace 27

4. Build 32
4.1 Build Option Settings 32
4.2 Build A Sample Project 35
4.3 Export Build Configuration Settings 36

5. Debug 37
5.1 Change Existing Debug Configurations 37
5.2 Create New Debug Configurations 42
5.3 Launch Bar 43
5.4 Basic Debugging Features 44
5.4.1 Breakpoints View 45
5.4.2 Expressions View 46
5.4.3 Registers View 48
5.4.4 Memory View 49
5.4.5 Disassembly View 51
5.4.6 Variables View 53
5.4.7 Eventpoints View 54
5.4.8 IO Registers View 57
5.4.9 Trace View 58
5.4.10 Memory Usage View 61

6. Help 65

1. General

Renesas e² studio is the Integrated Development Environment for Renesas embedded microcontrollers. e² studio is based on the industry-standard open-source Eclipse IDE framework and the C/C++ Development Tooling (CDT) project, covering build (editor, compiler, and linker control) and debug phases with an extended GNU Debug (GDB) interface support.

This chapter describes the system configuration and operating environment for e² 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.

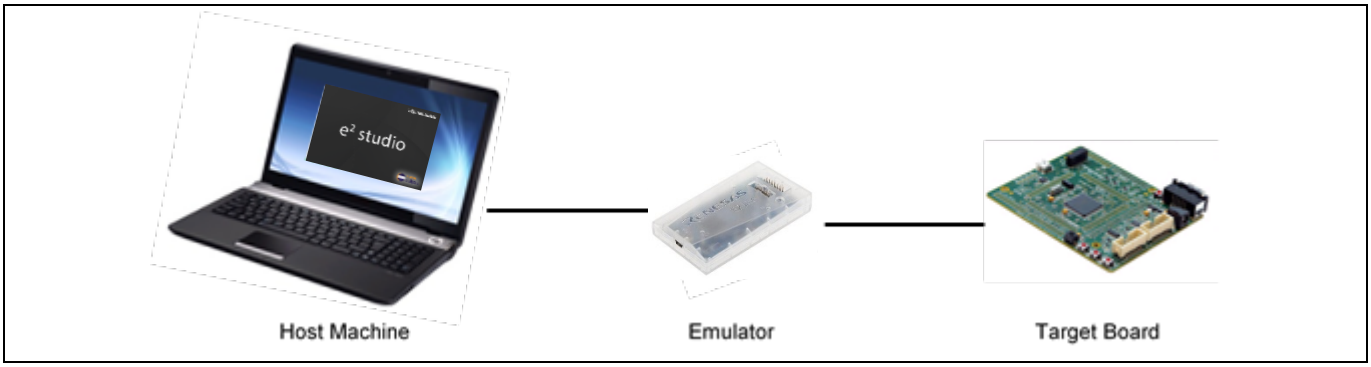


Figure 1-1 System configuration

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 2GB of free memory space.
Harddisk Capacity:	At least 2GB of free space
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.

Operating Environment:
Windows 8.1 (32/64-bit OS) and Windows 10 (32/64-bit OS). 64-bit OS is required for e² studio 2020-04 or later.

1.2.2 Supported Toolchains

Renesas C/C++ compiler package for RX family
Renesas C compiler package for RL78 family
GNURX Windows Toolchain (ELF)
GNURL78 Windows Toolchain (ELF)

Note1: Two types of packages (with CS+ and without IDE) for each CC-RX and CC-RL are available. Any of those can be applied to e² studio.

Note 2: For RH850, the e² studio can be used to debug load modules in the ELF/DWARF format which were built with the IAR Embedded Workbench from IAR Systems or the MULTI IDE from Green Hills Software.

1.3 Supported Emulator Device

E2 emulator Lite (RX, RL78)
E1 (RX, RL78, RH850)
E2 (RX, RL78, RH850)
E20 (RX)

1.4 Supported Simulator

Renesas Simulator (RX, RL78)
GDB Simulator (RH850)

2. Installation

The latest e² studio IDE installer package can be downloaded from Renesas website for free. Renesas now supports both 32-bit and 64-bit versions of e² studio.

Please check detailed information from https://www.renesas.com/e2studio. Note that 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 e² studio IDE.

e² studio installer can be used to upgrade e² studio as well as new installation. However, it does not support update between major versions such as from v5.x to v6.y, or from v6.m to v7.n.

Please uninstall the earlier versions before installation. Alternatively, install new e² studio into a new folder if you would like to keep earlier versions.

The detailed information is described below.

2.1 Installation of e² studio IDE (64-bit version)

1. Double-click on e² studio installer to invoke the e² studio installation wizard page. Click [Install].

   Note: If e² studio was installed in your PC, the option to modify, remove the existing version or install e² studio to a different location will be displayed.

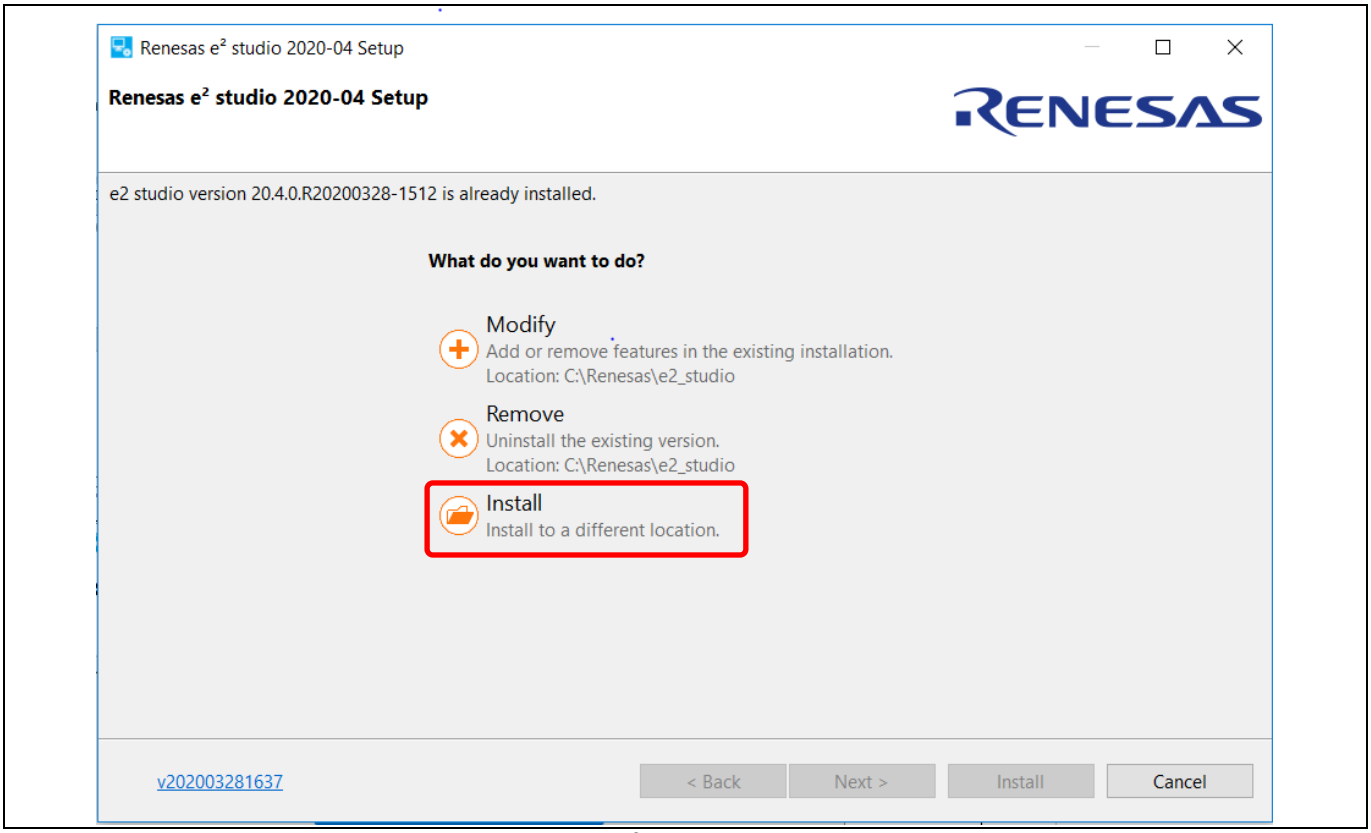


Figure 2-1 64-bit e² studio installation wizard

2. Welcome page

The install folder can be changed by clicking [Change…]. Click [Next] to continue.

Note1: If you would like to have multiple versions of e² studio, please specify new folder here.
Note2: Multi-byte characters cannot be used for e² studio installation folder name.

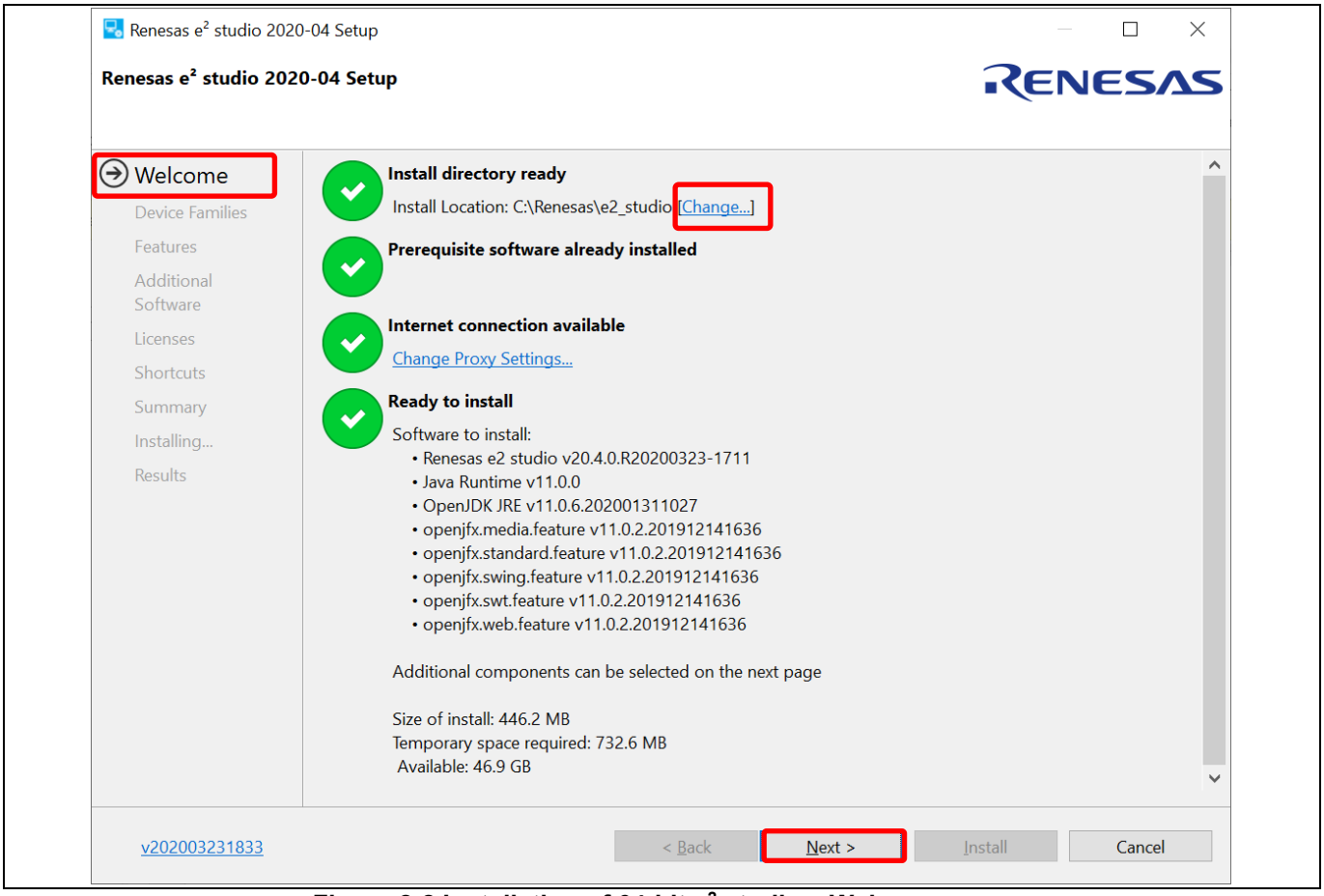


Figure 2-2 Installation of 64-bit e² studio – Welcome page

3. Device Families

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

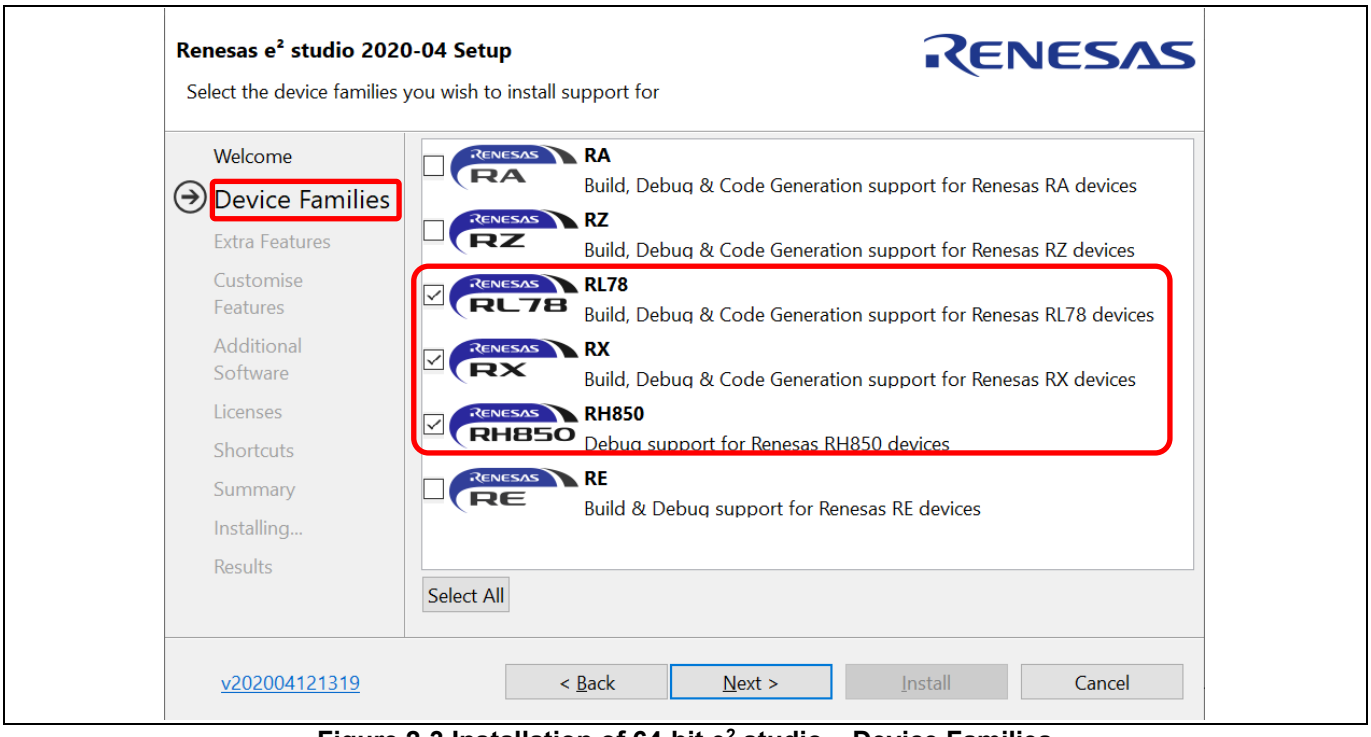


Figure 2-3 Installation of 64-bit e² studio – Device Families

4. Extra Features

Select Extra Features (i.e. Language packs, SVN & Git support, RTOS support…) to install.
For non-English language menu, please select Language packs at this step.

Click [Next] to continue.

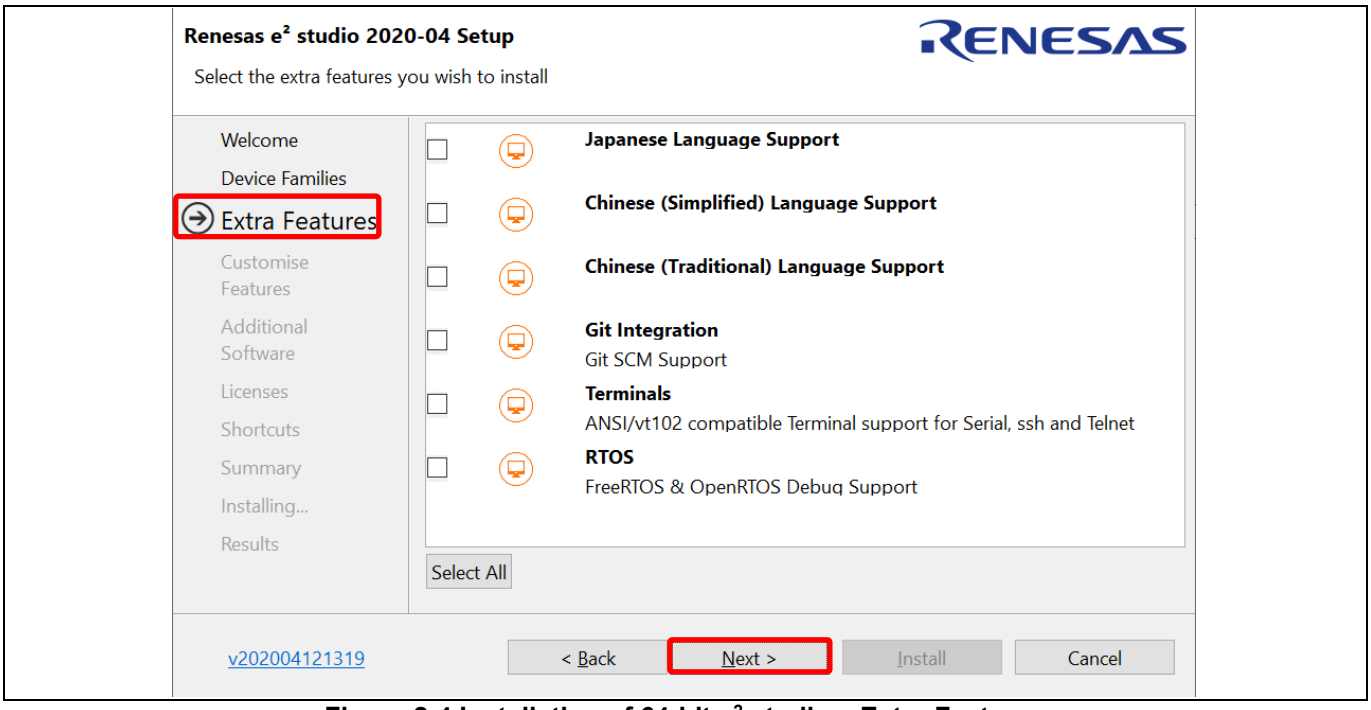


Figure 2-4 Installation of 64-bit e² studio – Extra Features

5. Customise Features

Select the components to install and click the [Next] button to continue.

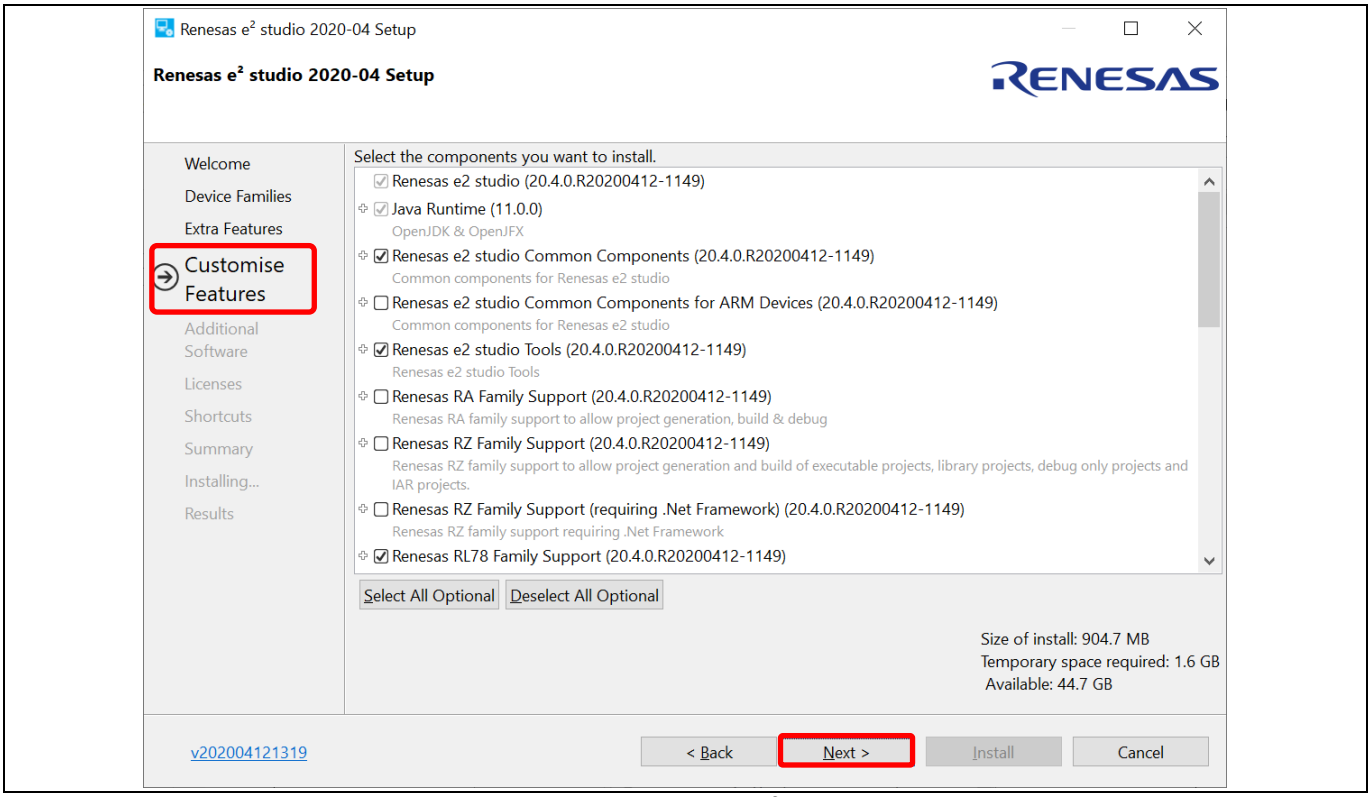


Figure 2-5 Installation of 64-bit e² studio – Features

6. Additional Software

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

Note: With no Internet access available, additional software installation can be skipped because software catalog cannot be downloaded. The additional software can be installed later.

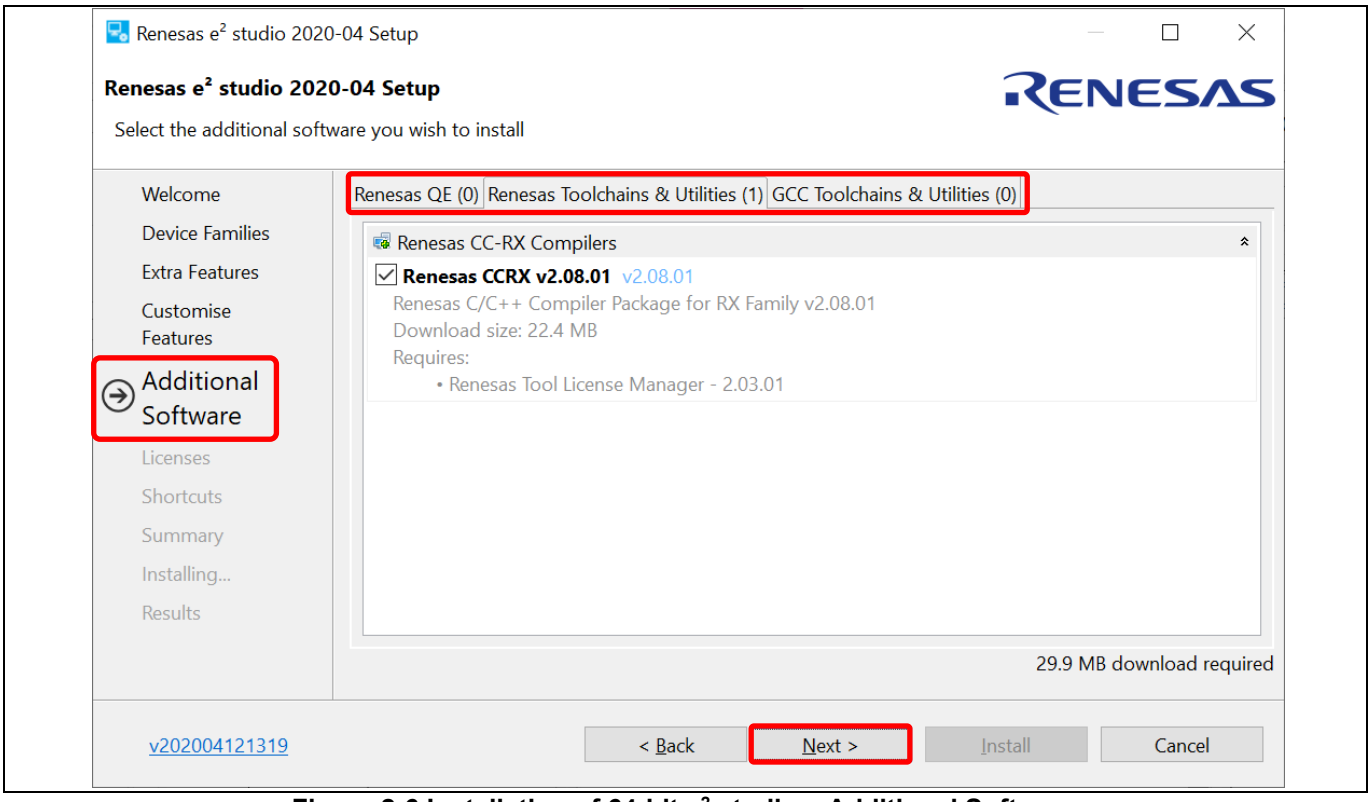


Figure 2-6 Installation of 64-bit e² studio – Additional Software

7. License Agreement

Read and accept the software license agreement. Click the [Next] button.
Please note that user must accept the license agreement, otherwise installation cannot be continued.

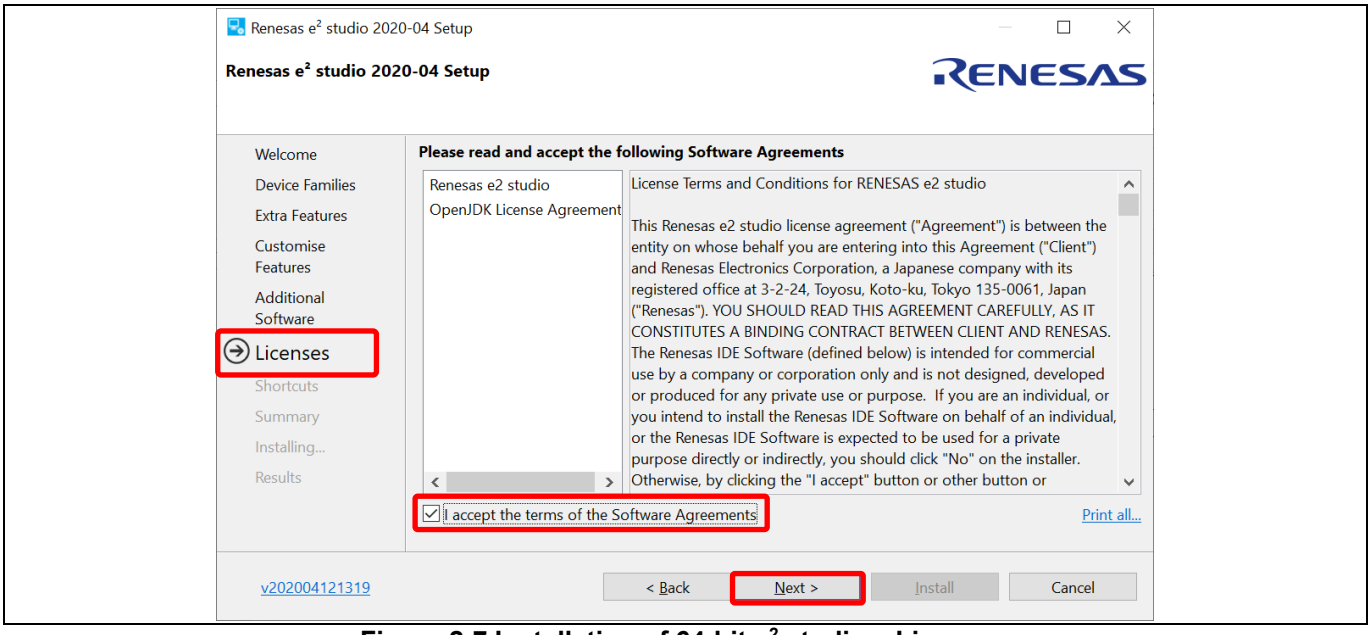


Figure 2-7 Installation of 64-bit e² studio – Licenses

8. Shortcuts

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

Note: If e² studio was installed in another location, it is recommended to rename to distinguish from the other e² studio(s).

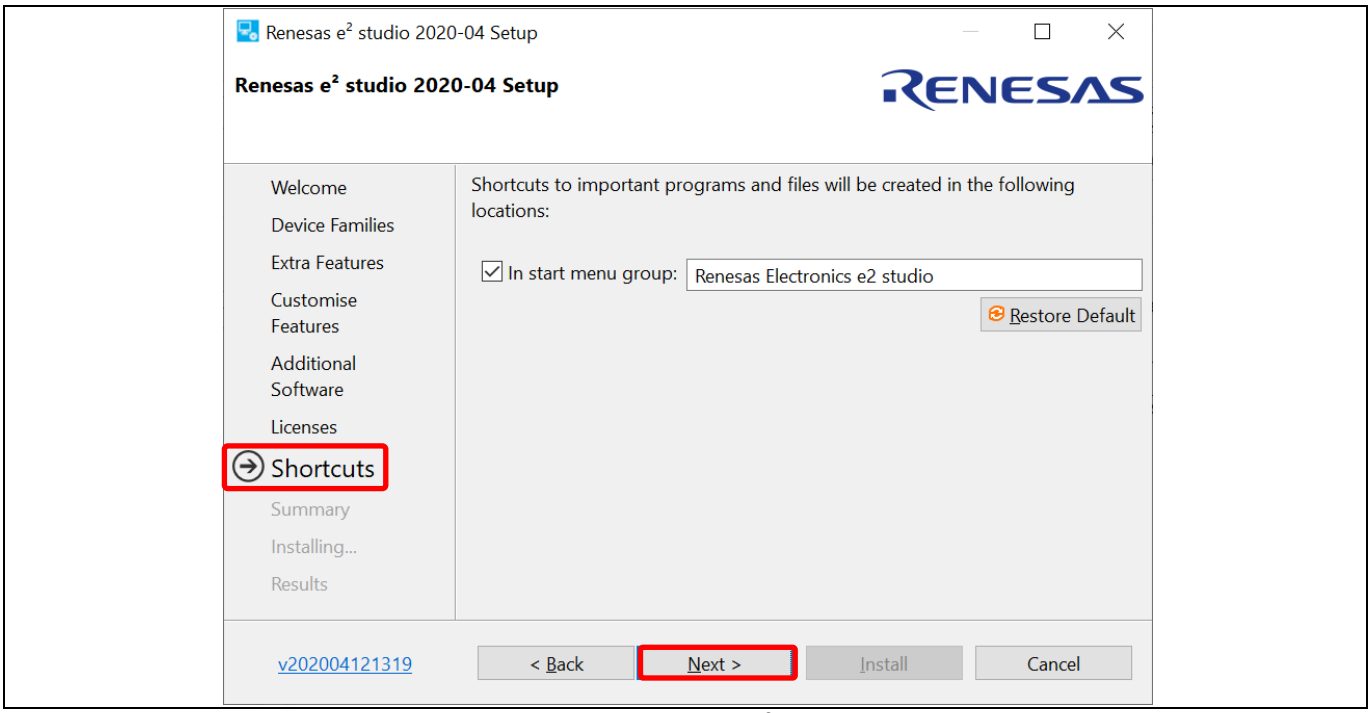


Figure 2-8 Installation of 64-bit e² studio – Shortcuts

9. Summary

Components list to be installed is shown. Please confirm the contents and click the [Install] button to install the Renesas e² studio IDE.

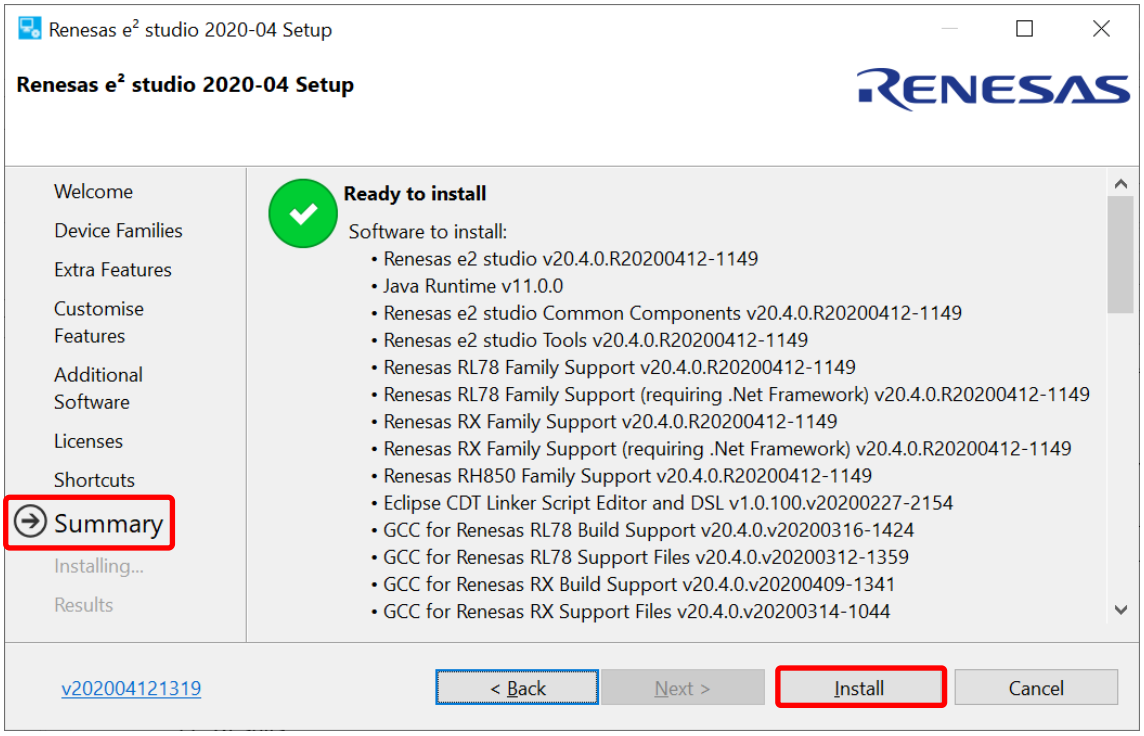


Figure 2-9 Installation of 64-bit e² studio – Summary

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

    Installation results are listed here. Please note if any errors are shown.

    Click [OK] button to complete the installation.

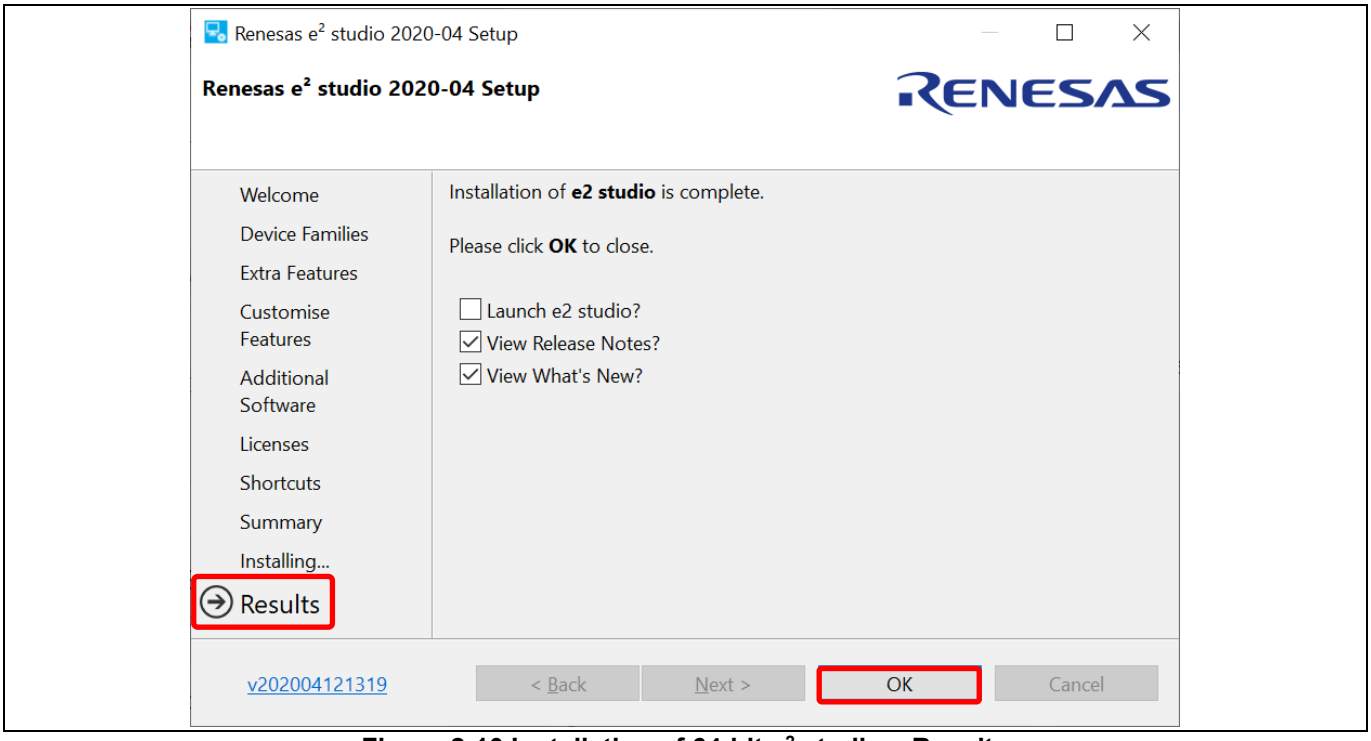


Figure 2-10 Installation of 64-bit e² studio – Results

2.2 Installation of e² studio IDE (32-bit version)

1. Double-click on e² studio installer to invoke the e² studio installation wizard page. Click [Install].

   Note: If e² studio was installed in your PC, the option to modify, remove the existing version or install e² studio to a different location will be displayed. If you would like to have multiple versions of e² studio, please specify “Install to a different location” here. Click [Next] to continue.

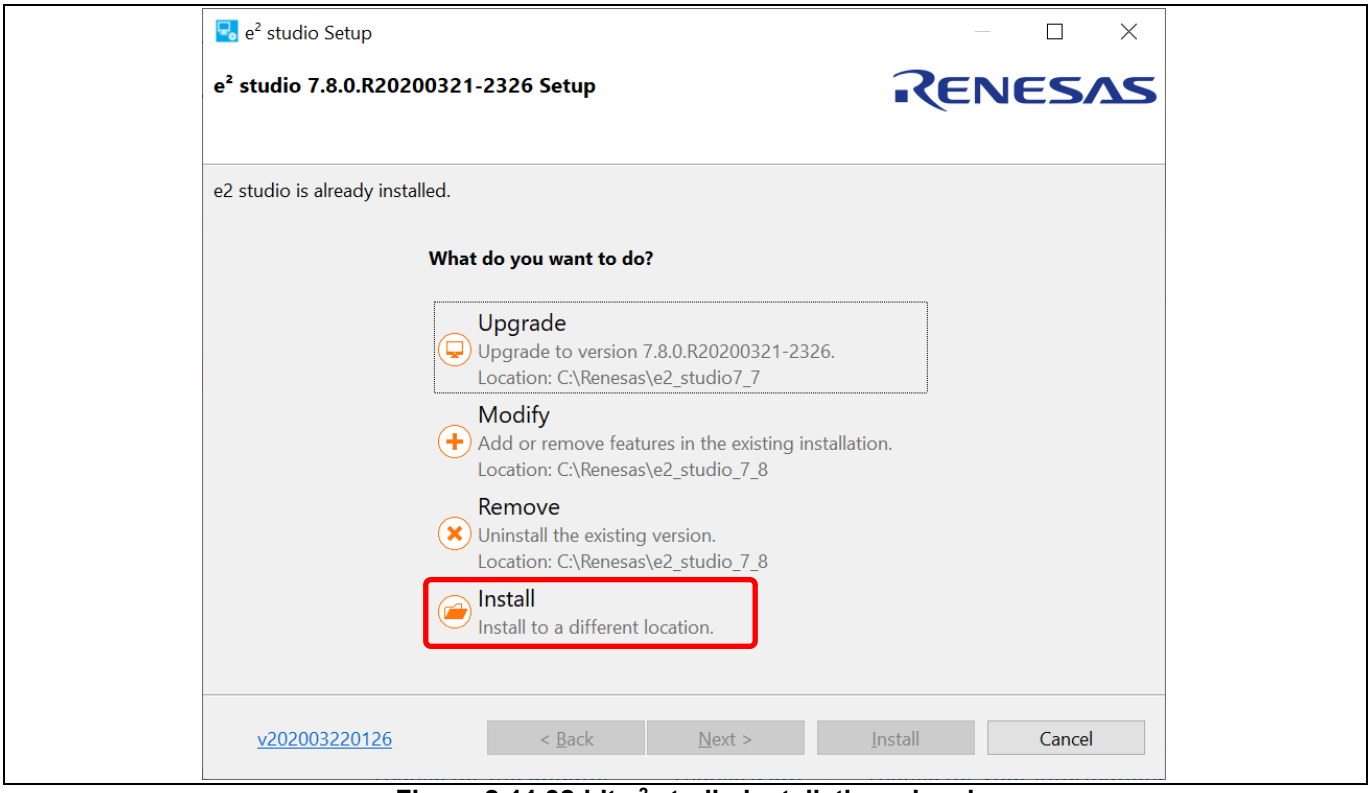


Figure 2-11 32-bit e² studio installation wizard

2. Welcome page

   Click [Next] to continue.

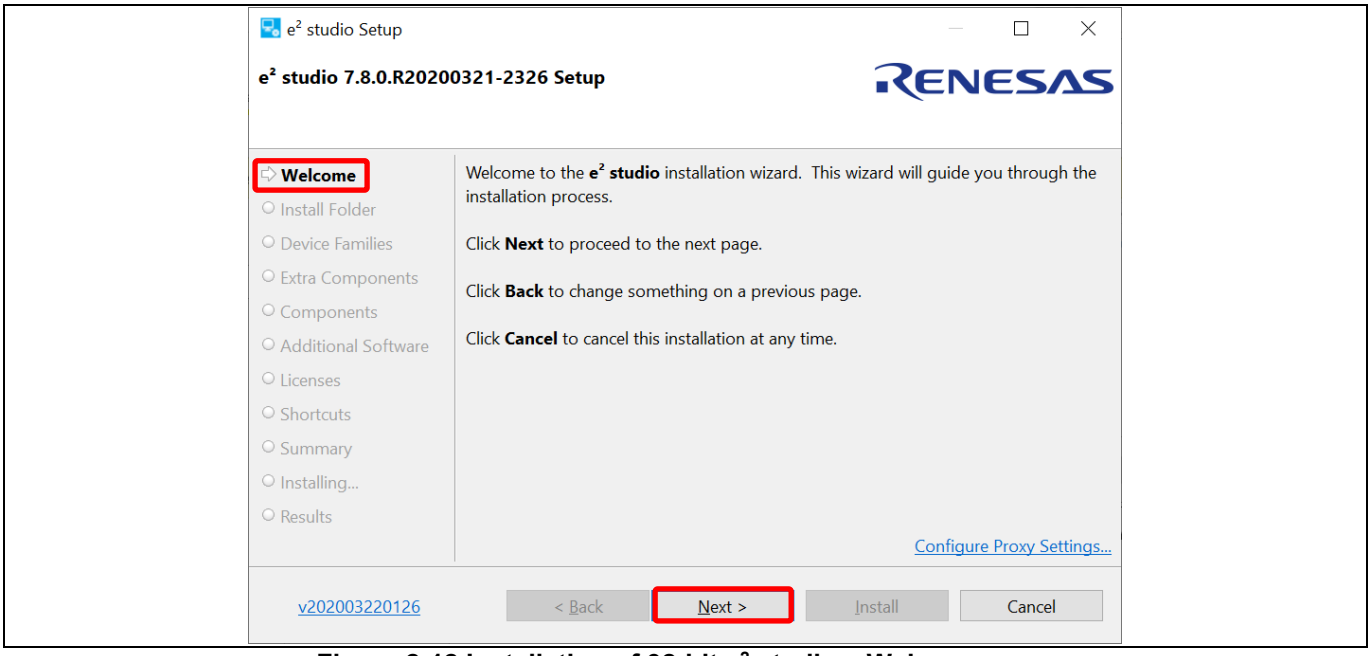


Figure 2-12 Installation of 32-bit e² studio – Welcome page

3. Install Folder

The default installation location is set to: “C:\Renesas\e2_studio”. Enter folder name directly in the textbox or click [Browse…] button to modify it. Click [Next] to continue.

Note1: If you would like to have multiple versions of e² studio, please specify new folder here.
Note2: Multi-byte characters cannot be used for e² studio installation folder name.

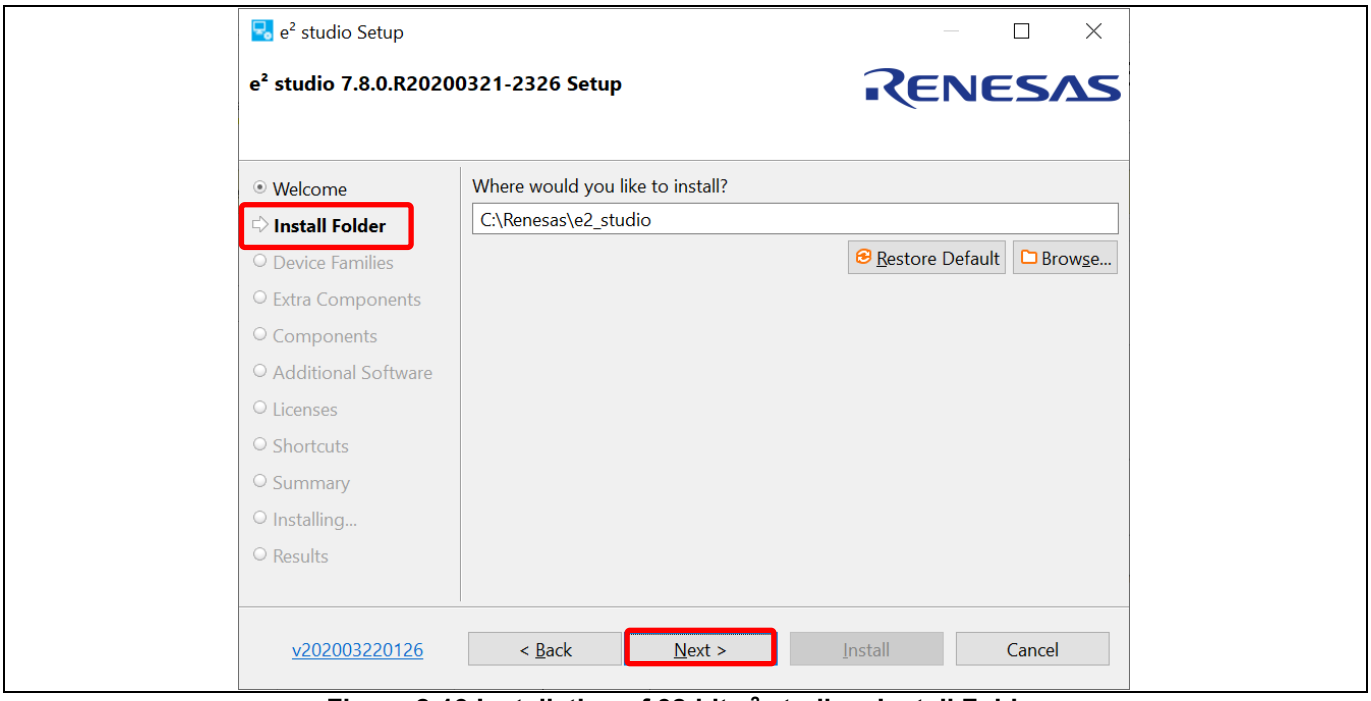


Figure 2-13 Installation of 32-bit e² studio – Install Folder

4. Device Families

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

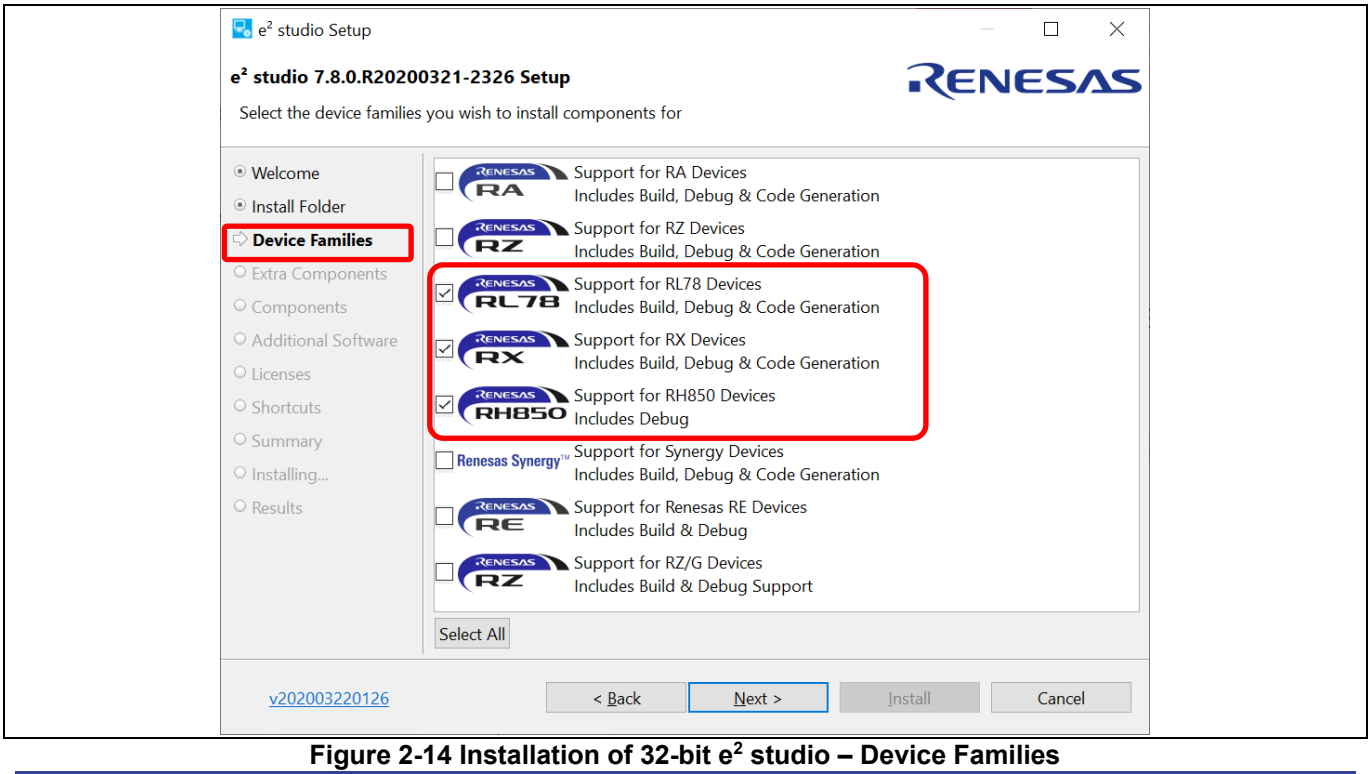


Figure 2-14 Installation of 32-bit e² studio – Device Families

5. Extra Components

Select Extra Components (i.e. Language packs, SVN & Git support, RTOS support…) to install.
For non-English language users, please select Language packs at this step.

Click the [Next] button to continue.

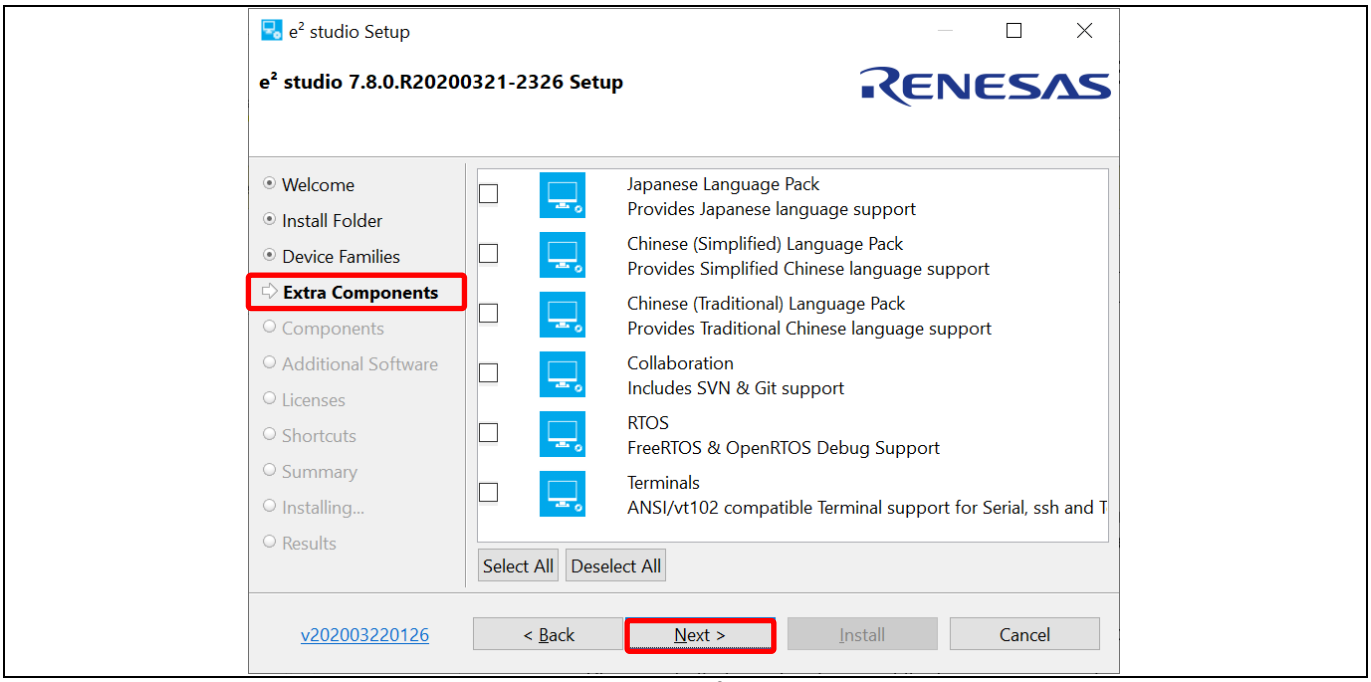


Figure 2-15 Installation of 32-bit e² studio – Extra Components

6. Components

Select the components to install and click the [Next] button to continue.

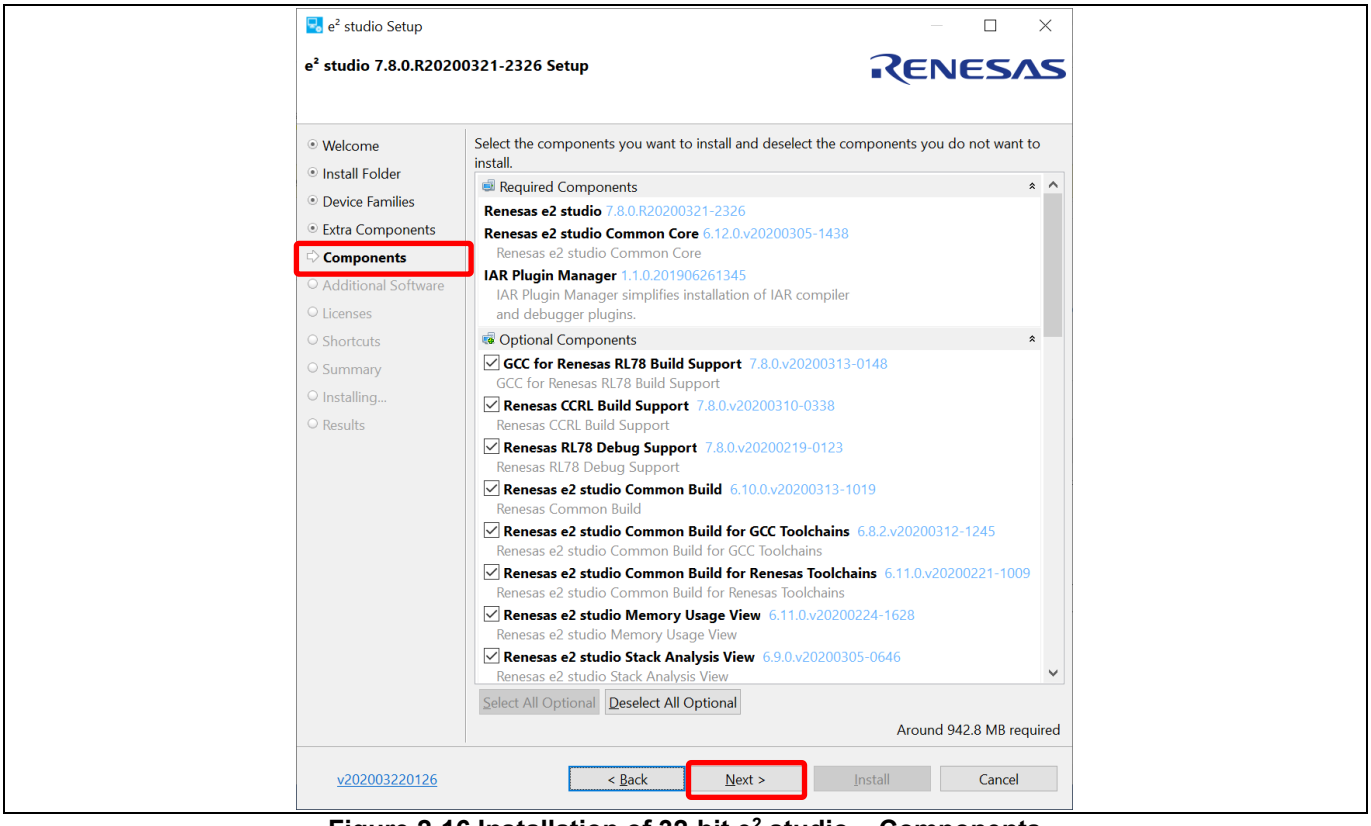


Figure 2-16 Installation of 32-bit e² studio – Components

7. Additional Software

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

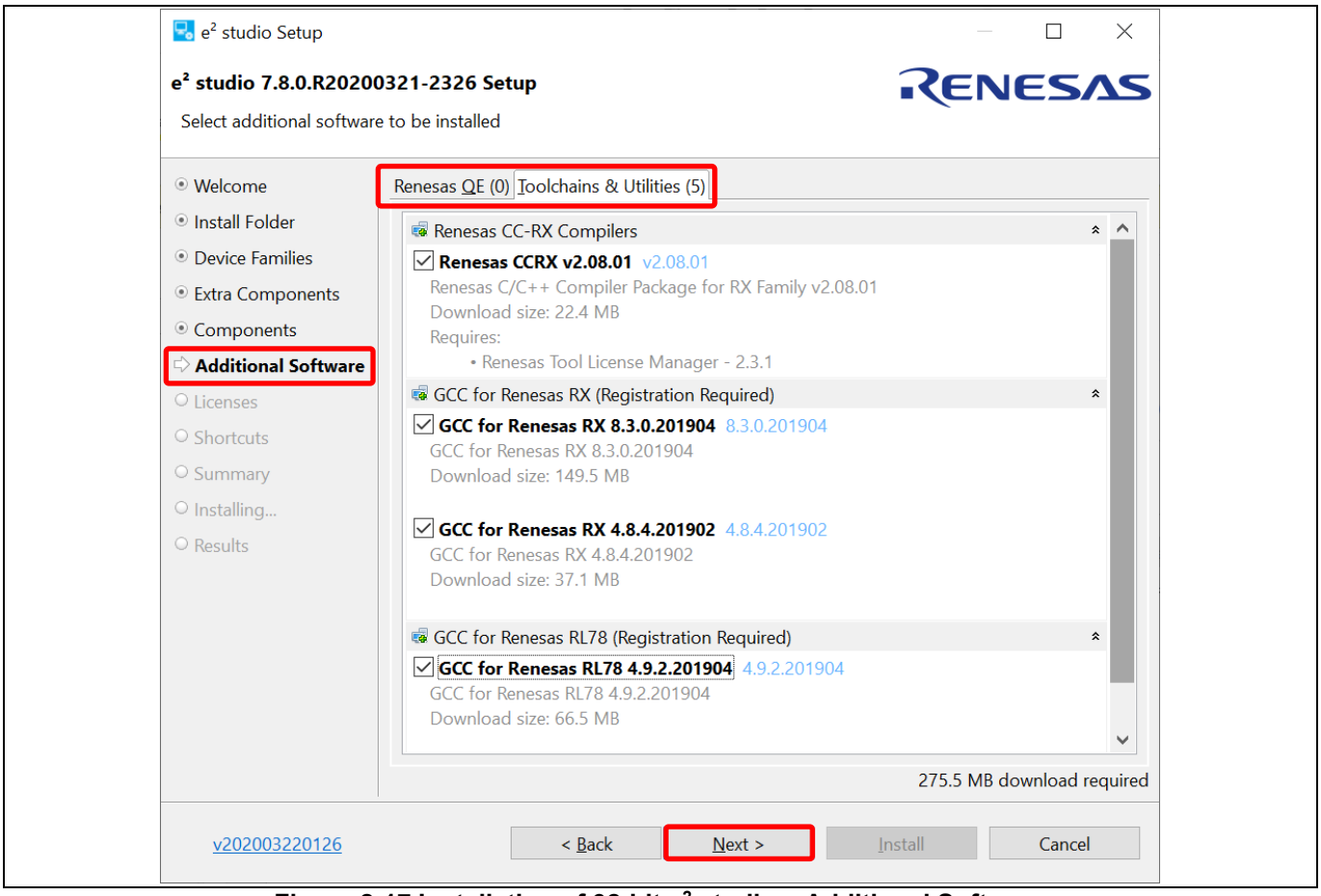


Figure 2-17 Installation of 32-bit e² studio – Additional Software

8. License Agreement, Shortcuts, Summary, Installing… and Results

The steps for these pages are the same as installation of e² studio 64-bit version.

Please refer to Chapter 2.1, step 7 onwards.

2.3 Un-installation of e2 studio IDE

User can uninstall e2 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 “e2 studio” and click the [Uninstall] button.
3. Click the [Uninstall] button again to confirm the deletion of e2 studio.

   At the end of the un-installation, e2 studio IDE will be deleted from the installed location and Windows shortcuts menu are removed.

Note: If you have installed e2 studio at multiple locations, you may not find the uninstaller in “Apps & features” of Control Panel. In such cases, launch e2 studio uninstaller located at:
`{e2 studio installed folder}/uninstall/`**`uninstall.exe`**.

2.4 Update e² studio plugins

Following procedure describes how to update existing plugins, **not to install new features**. Please use e² studio installer to upgrade to 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] pane.

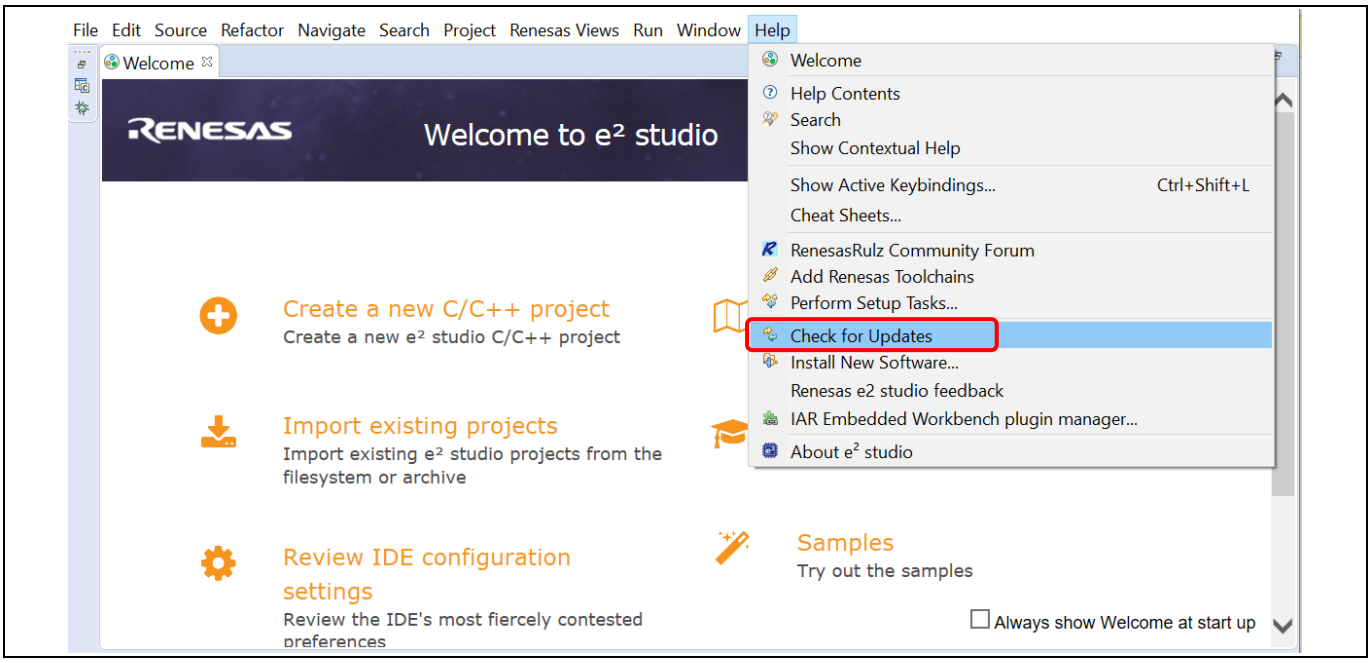


Figure 2-18 [Check for Updates] Menu

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

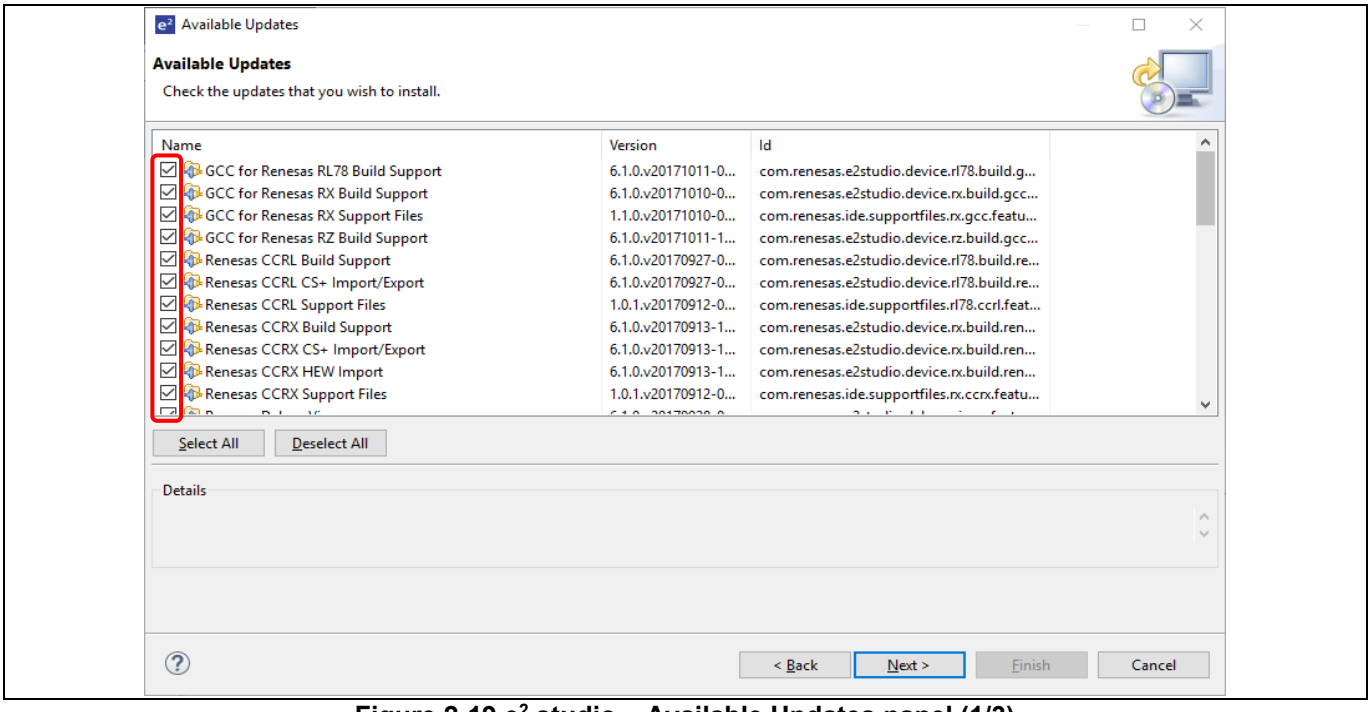


Figure 2-19 e² studio – Available Updates panel (1/3)

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

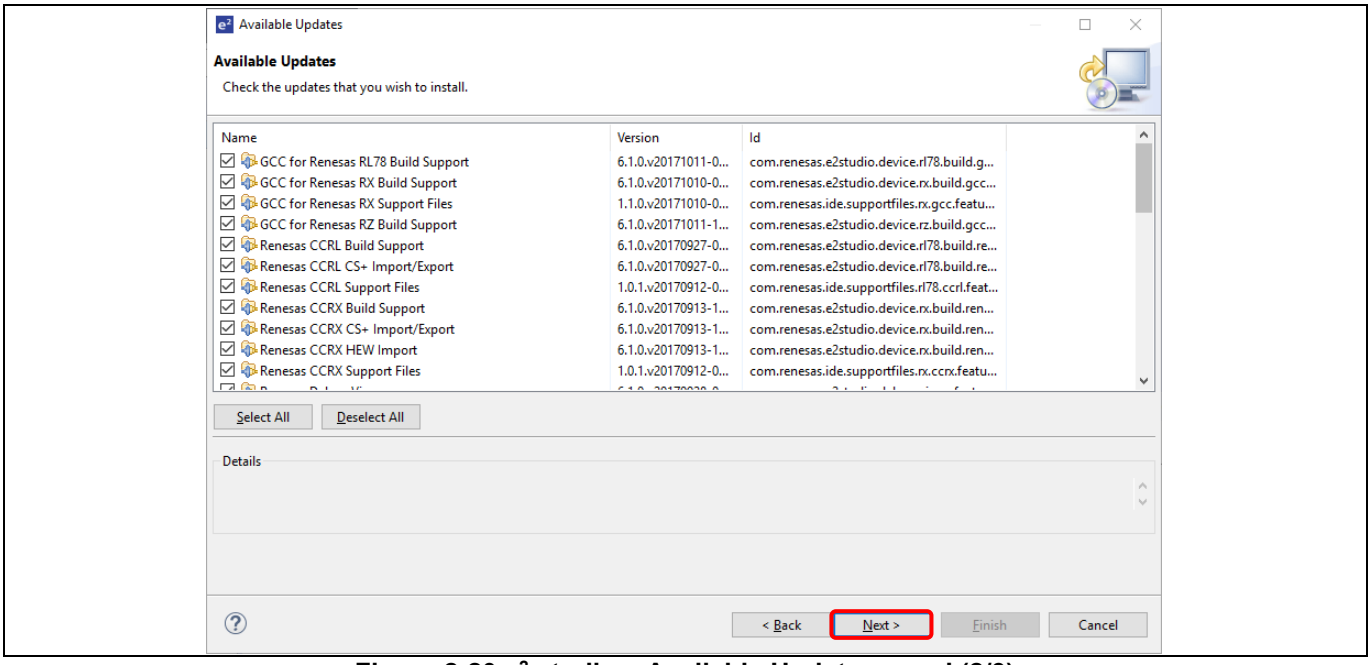


Figure 2-20 e² studio – Available Updates panel (2/3)

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

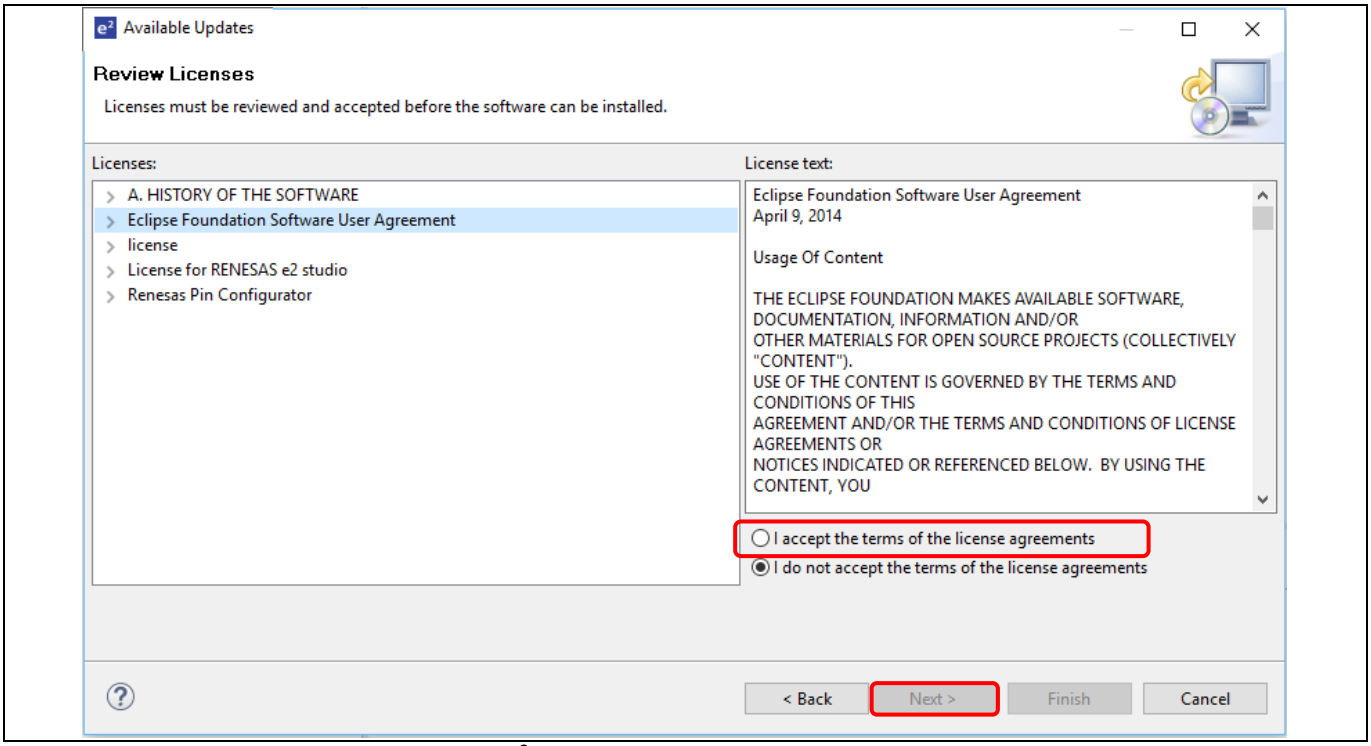


Figure 2-21 e² studio – Available Updates panel (3/3)

5. Click the menu [Help] → [About e² studio] → [Installation Details] to confirm the updated version.

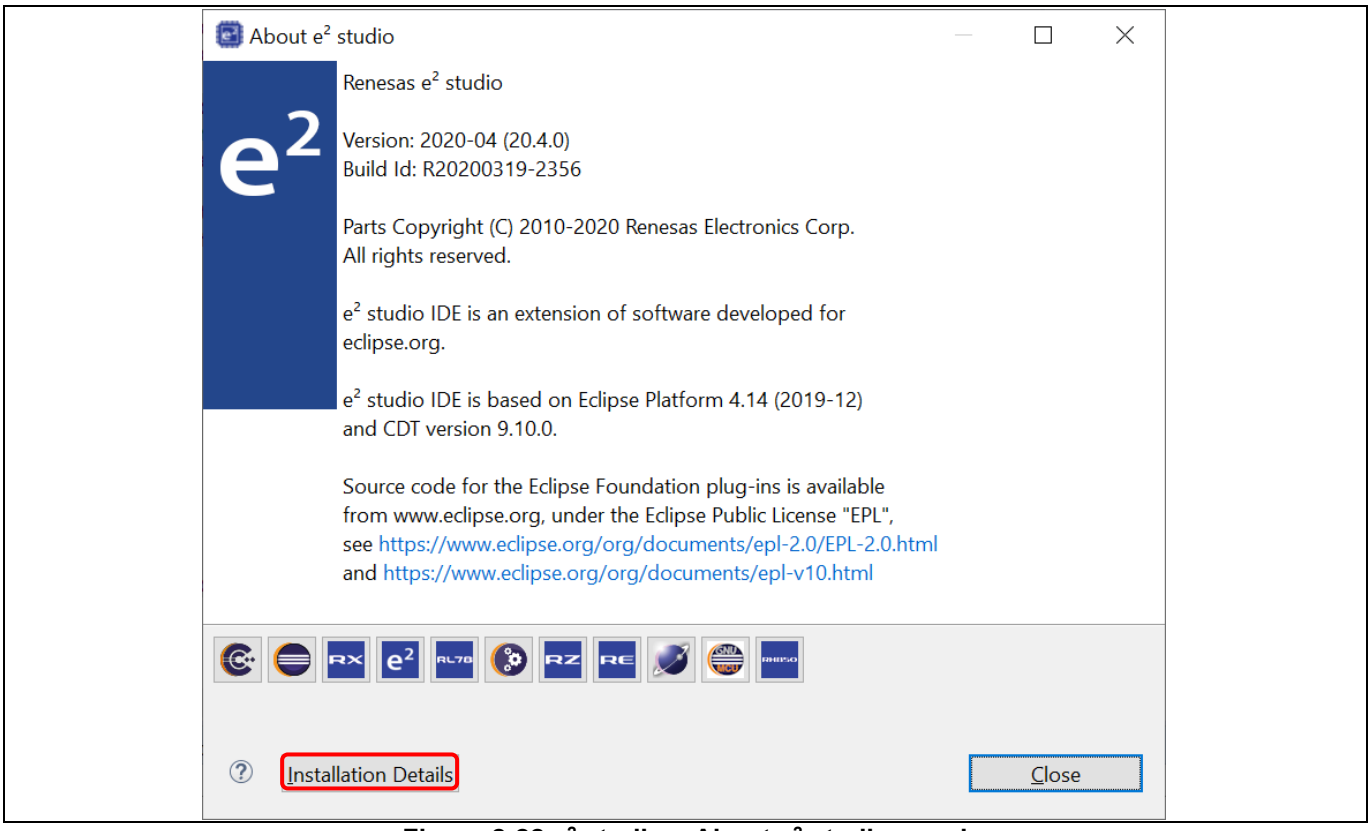


Figure 2-22 e² studio – About e² studio panel

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 e² 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 e² studio version to a different folder.
   Please choose any of the following options:

   [Upgrade] - Minor version upgrade of e² studio

   [Modify] - Add plugins into the same version of e² studio

   [Delete] - Uninstall e² studio

   [Install] - Install e² studio to a new location

Then click [Next] to proceed.

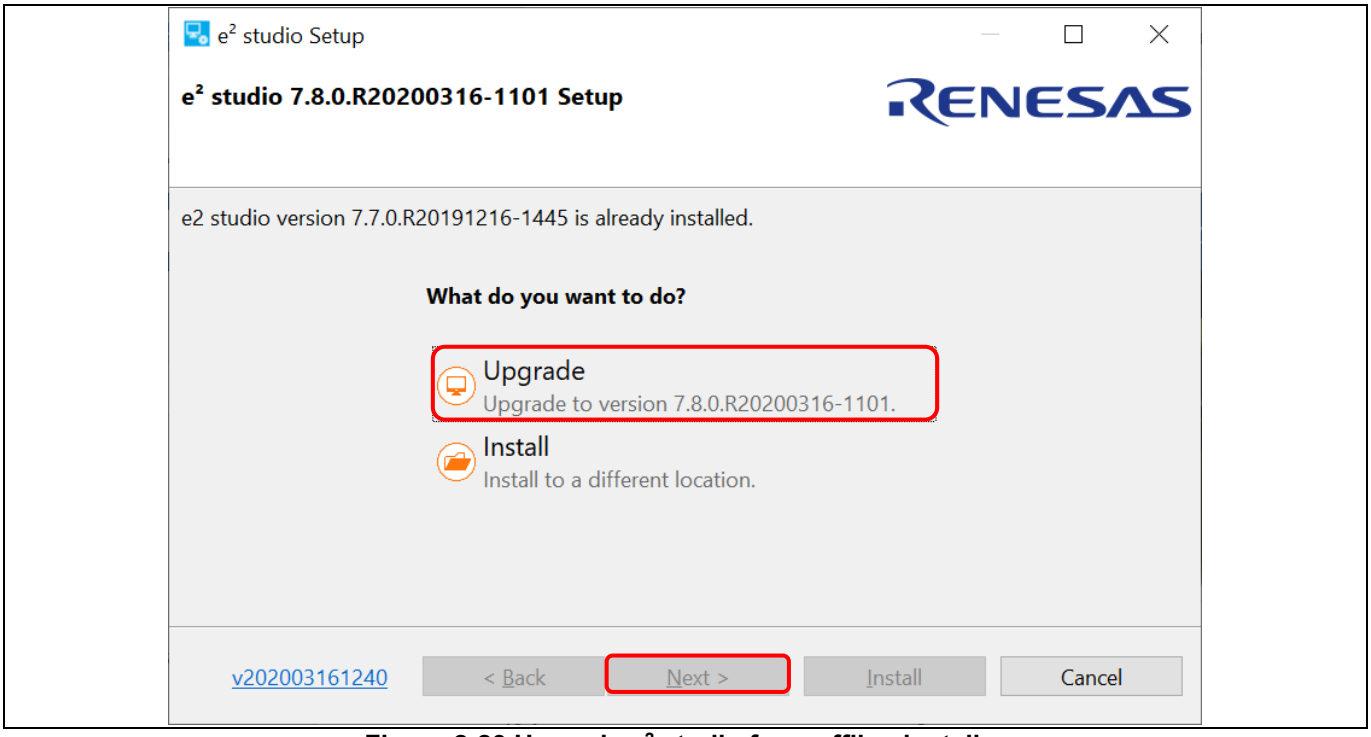


Figure 2-23 Upgrade e² studio from offline installer

Note: Please do not apply "Upgrade" or "Modify" if major versions are different between installer and the installed e² studio.

3. Follow the steps shown in Section 2.1 or 2.2 to install e² studio IDE. Step (3) Install Folder is skipped since Upgrade option will use the same destination folder as existing e² studio.

2.5 Installation of Compiler Package

e² studio installer could install compiler packages automatically during e² studio installation with valid Internet connection. However, in situation where Internet connection is not available during e² studio installation, compiler packages can be installed later from compiler package installation files from the web site shown below.

Compiler packages are available at each distribution sites. Please follow the installation guides at the download pages.

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 e² studio.

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

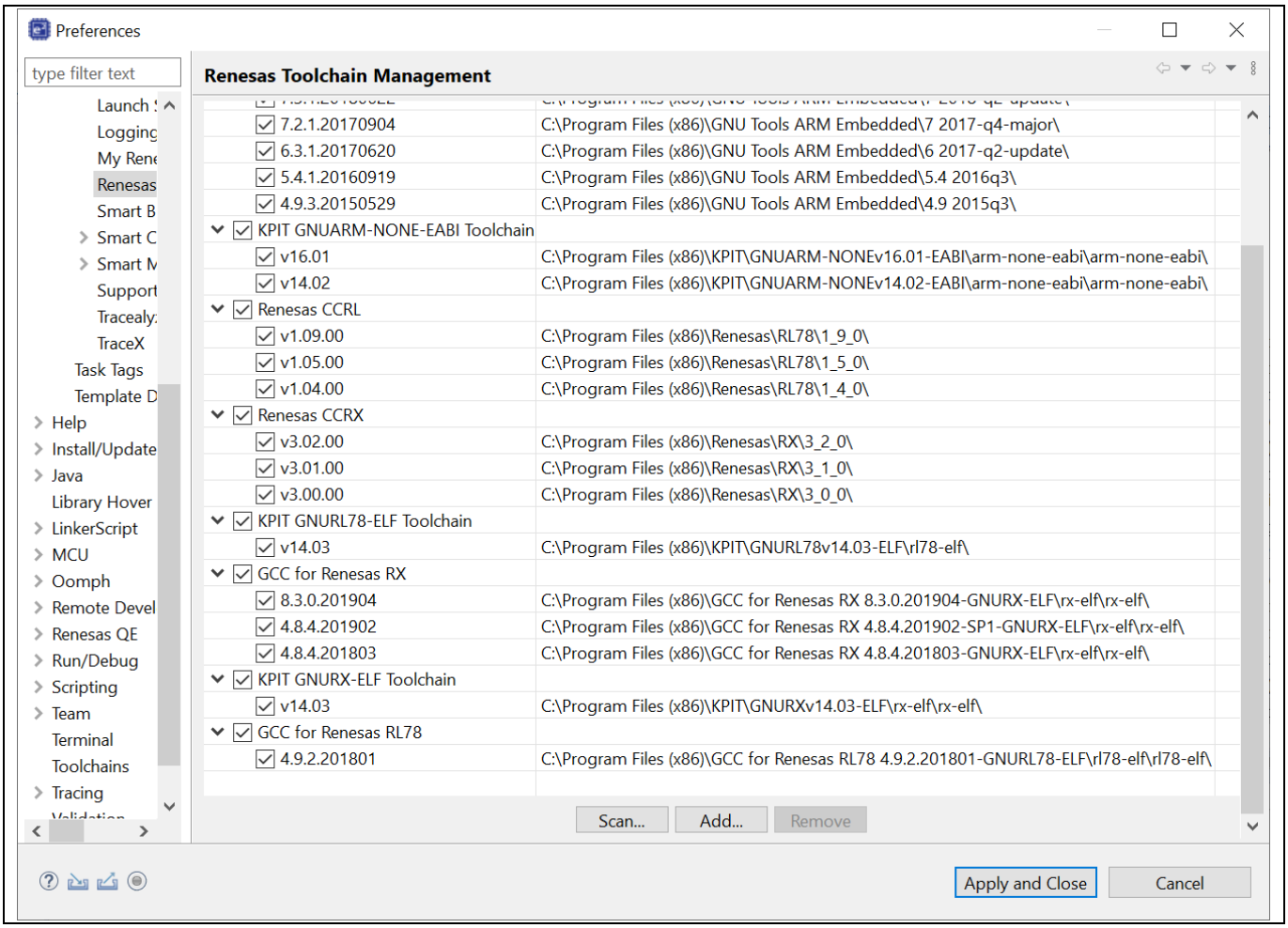


Figure 2-24 Toolchain management

3. Project generation

In e² studio, "Project" is a basic unit to perform build and debug operations.
This chapter describes the creation of new project and import of existing e² studio project, High-performance Embedded Workshop IDE (described as “HEW” below) project and CS+ project to e² studio IDE.

Note: 1. To install and use the e² studio on your PC, you must install the compiler package provided separately.

2. Multi-byte characters cannot be used for e² studio installation folder name, project name and its folder, and source file name.

3.1 New Project Generation

To create a new project, invoke e² 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.

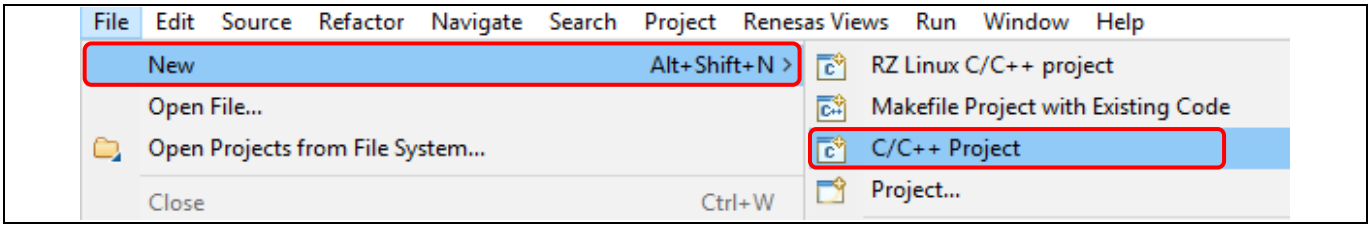


Figure 3-1 Open new project creation wizard

2. Select template for the new project (For e.g., Renesas RX: “Renesas CC-RX C/C++ Executable Project”). If the target device family or the toolchain were not listed, you may need to run e² studio installer to add "Build/Debug support plugins" for the target device family. Click [Next] to proceed.

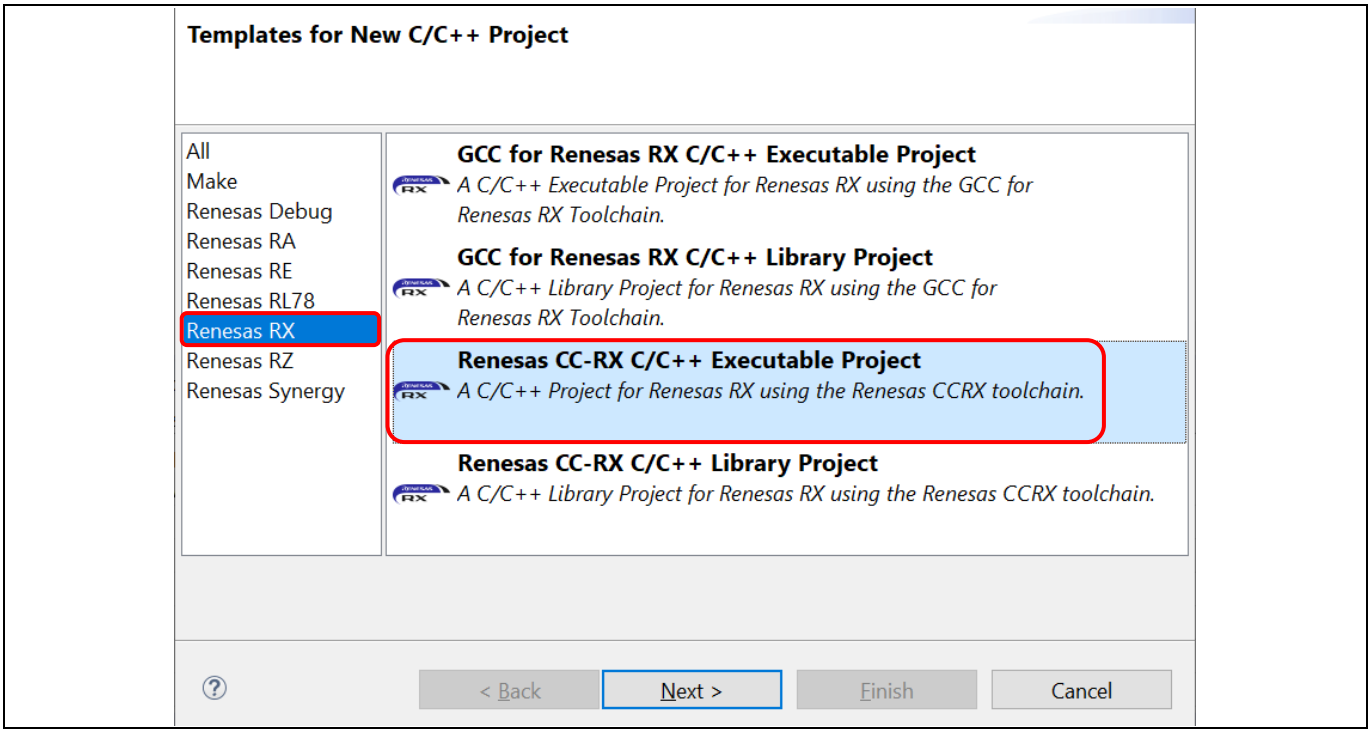


Figure 3-2 New Project Creation Wizard (1/6)

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

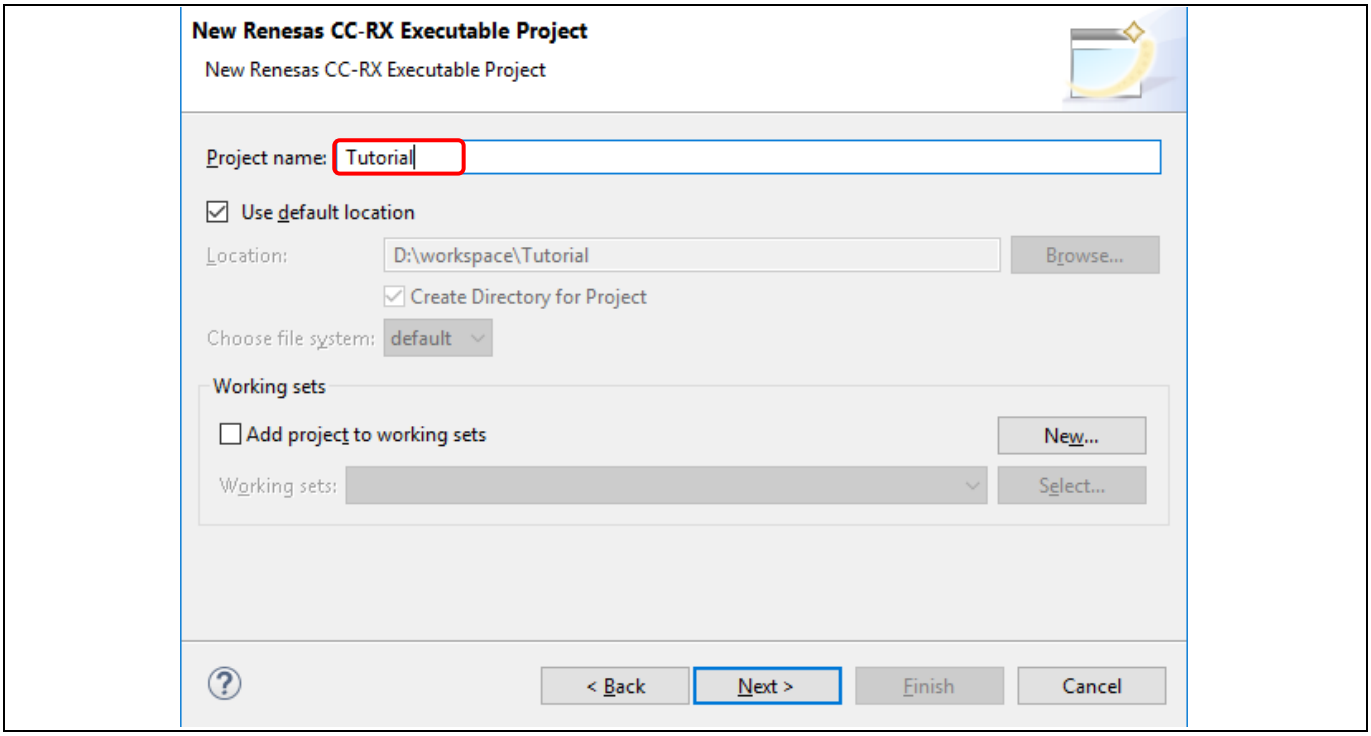


Figure 3-3 New Project Creation Wizard (2/6)

4. Select Language, Toolchain, Toolchain Version, RTOS, RTOS Version, Target Device and Configurations. Click [Next] to proceed.

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

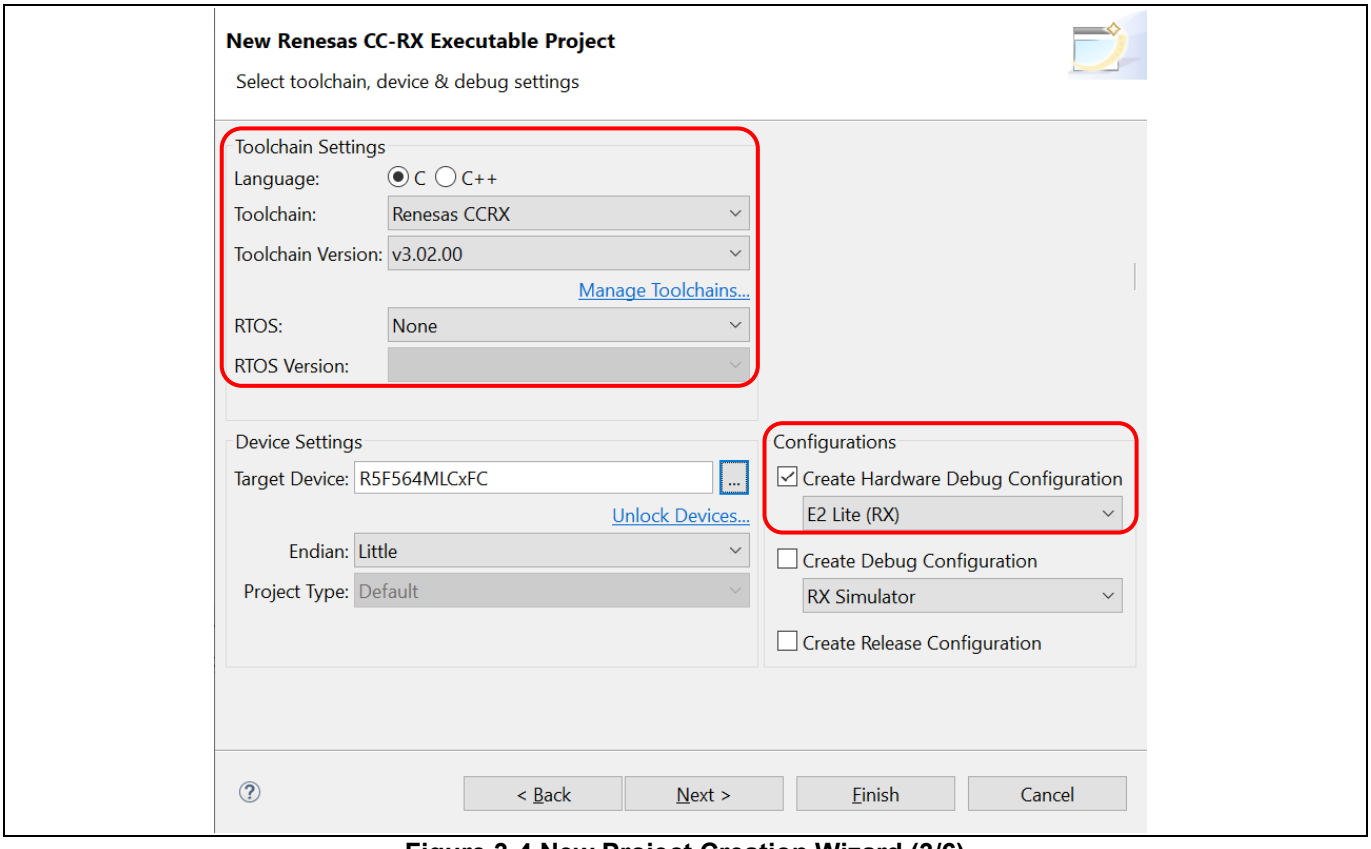


Figure 3-4 New Project Creation Wizard (3/6)

5. Coding Assistant feature can be applied if necessary. Click [Next] to proceed.

   Note:

   - *Peripheral Code Generator* (CG) 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* provides drivers and codes higher layer than CG, such as communication protocol stack and sample application programs using peripheral functions. All FIT modules are interchangeable because they are implemented with common interfaces.
   - *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.

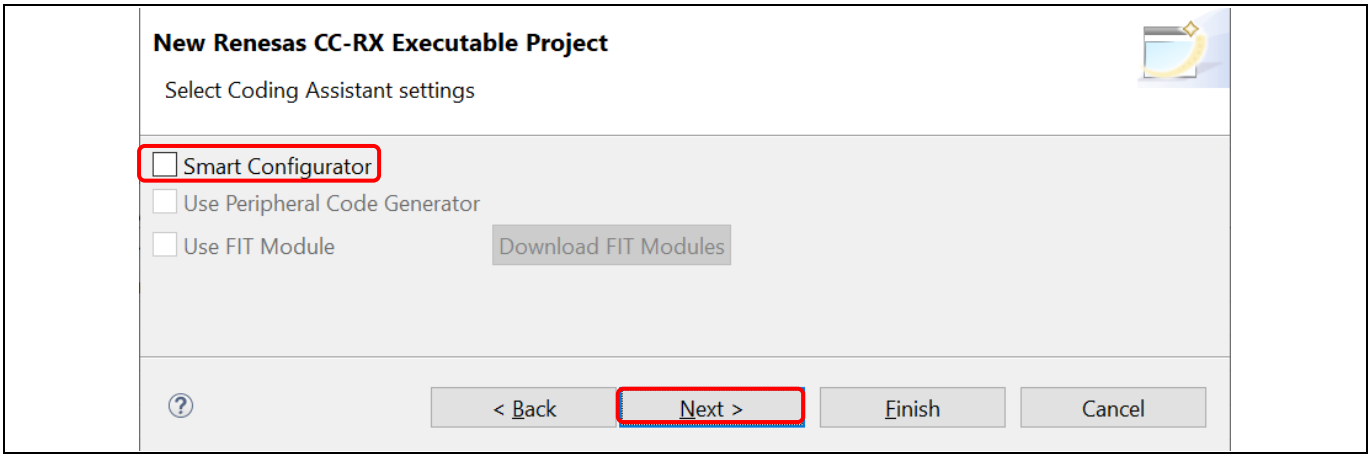


Figure 3-5 New Project Creation Wizard (4/6)

6. Keep the “Use Renesas Debug Virtual Console” unchecked and click [Next] to proceed.

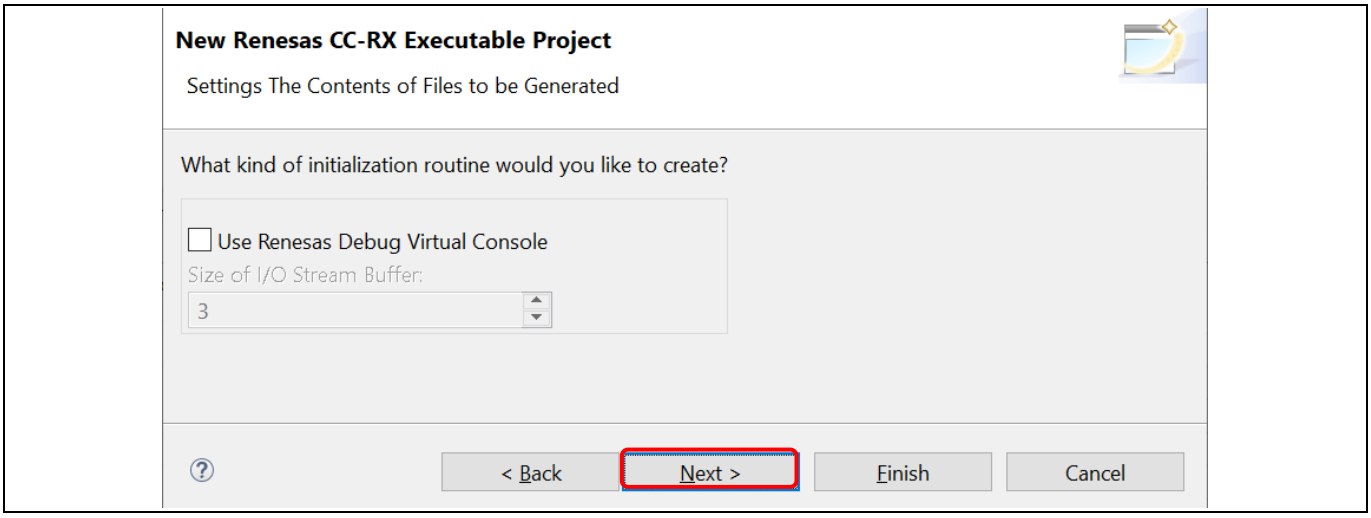


Figure 3-6 New Project Creation Wizard (5/6)

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

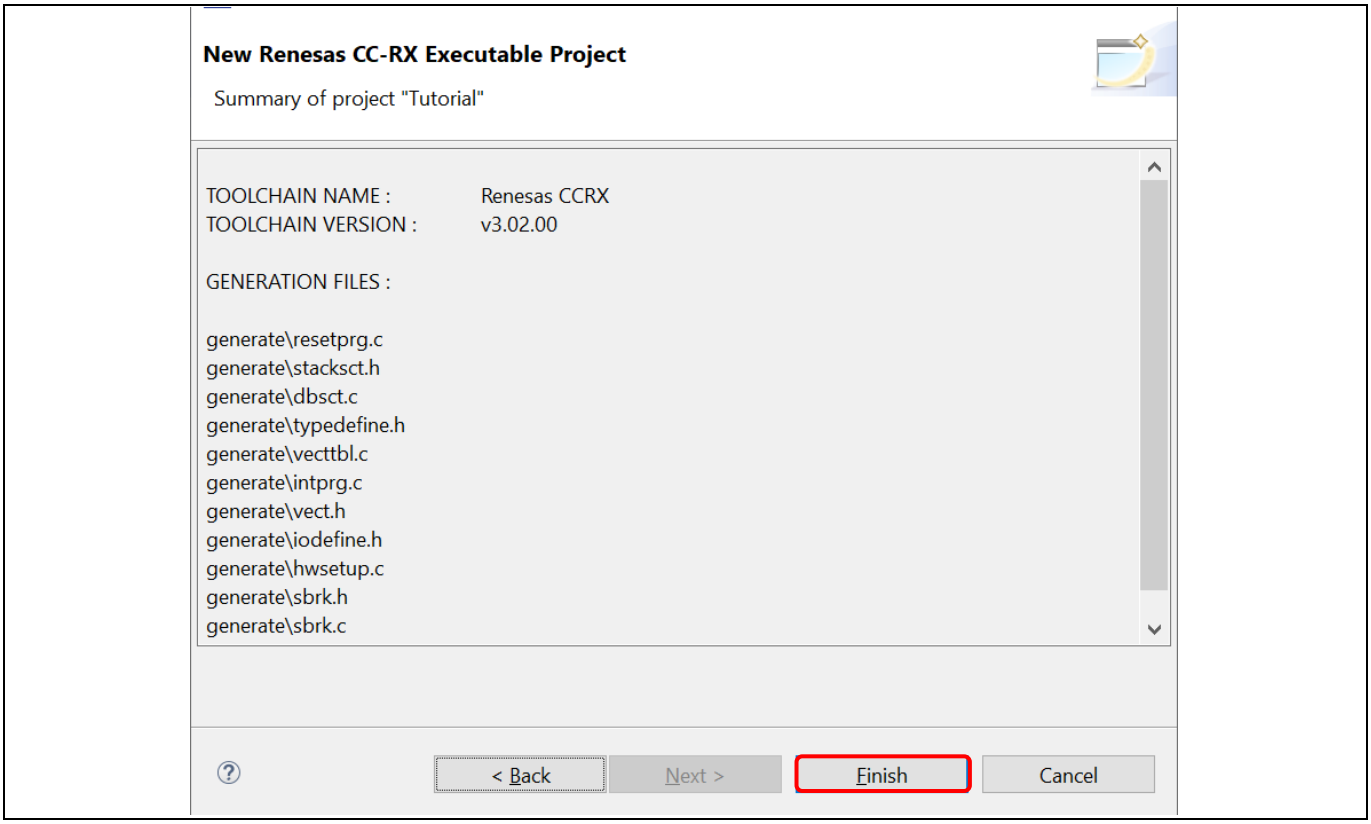


Figure 3-7 New Project Creation Wizard (6/6)

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

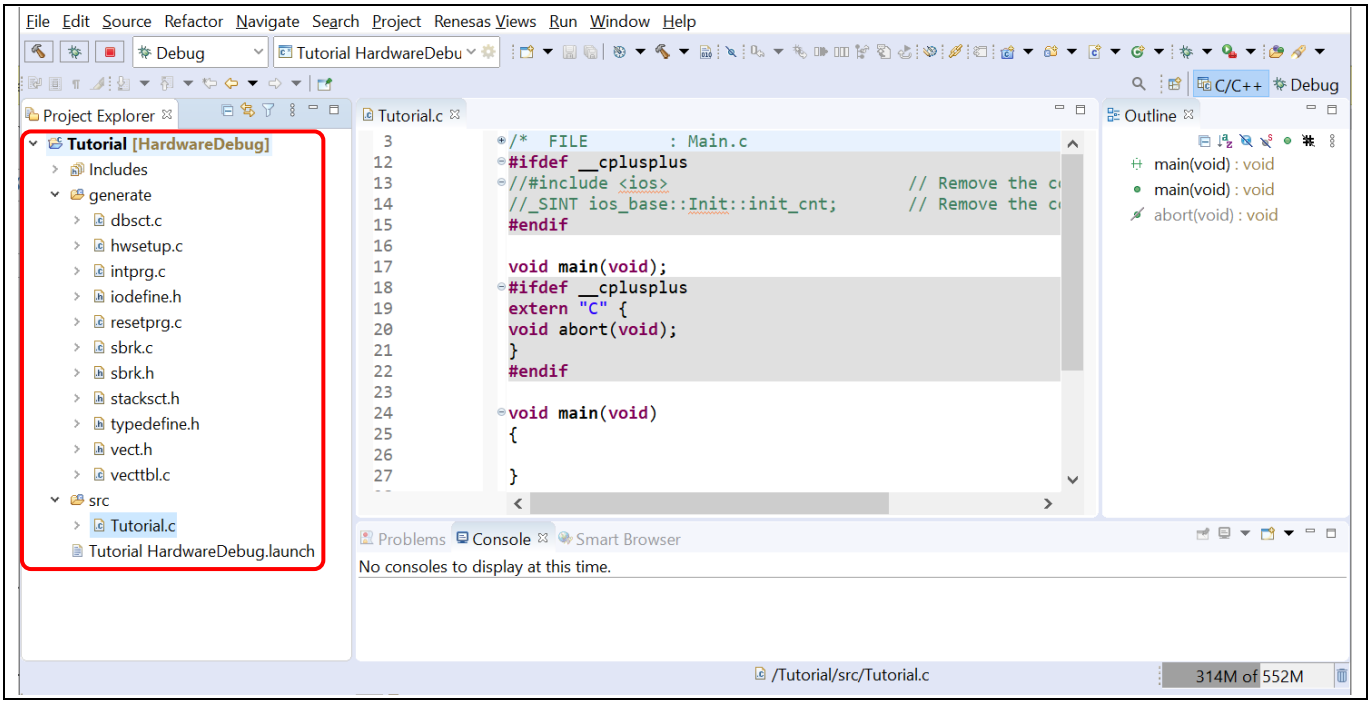


Figure 3-8 New C Project Created

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 New Debug Only Project Generation (RH850)

Creating a debug only project allows user to debug an existing executable which user has already built. This feature will automatically create a project and debug configuration for user.

Note: Only the debugging functions of the e² studio can be used with products of the RH850 family. The e² studio can be used to debug load modules in the ELF/DWARF format which were built with the IAR Embedded Workbench from IAR Systems or the MULTI IDE from Green Hills Software.

To create a debug only project (RH850 family is supported),

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

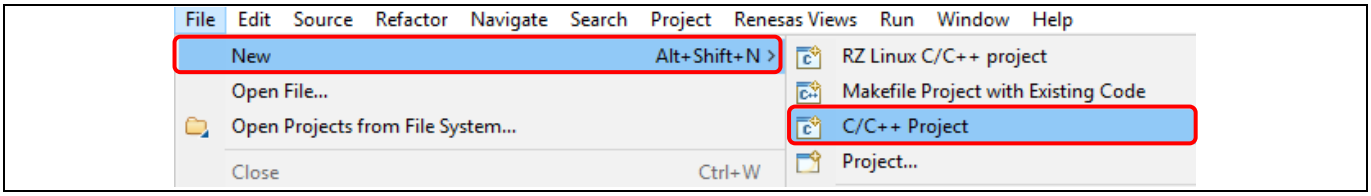


Figure 3-9 Open new project creation wizard

2. Select template for the new project: [Renesas Debug] → “Renesas Debug Only Project”. Click [Next] to proceed.

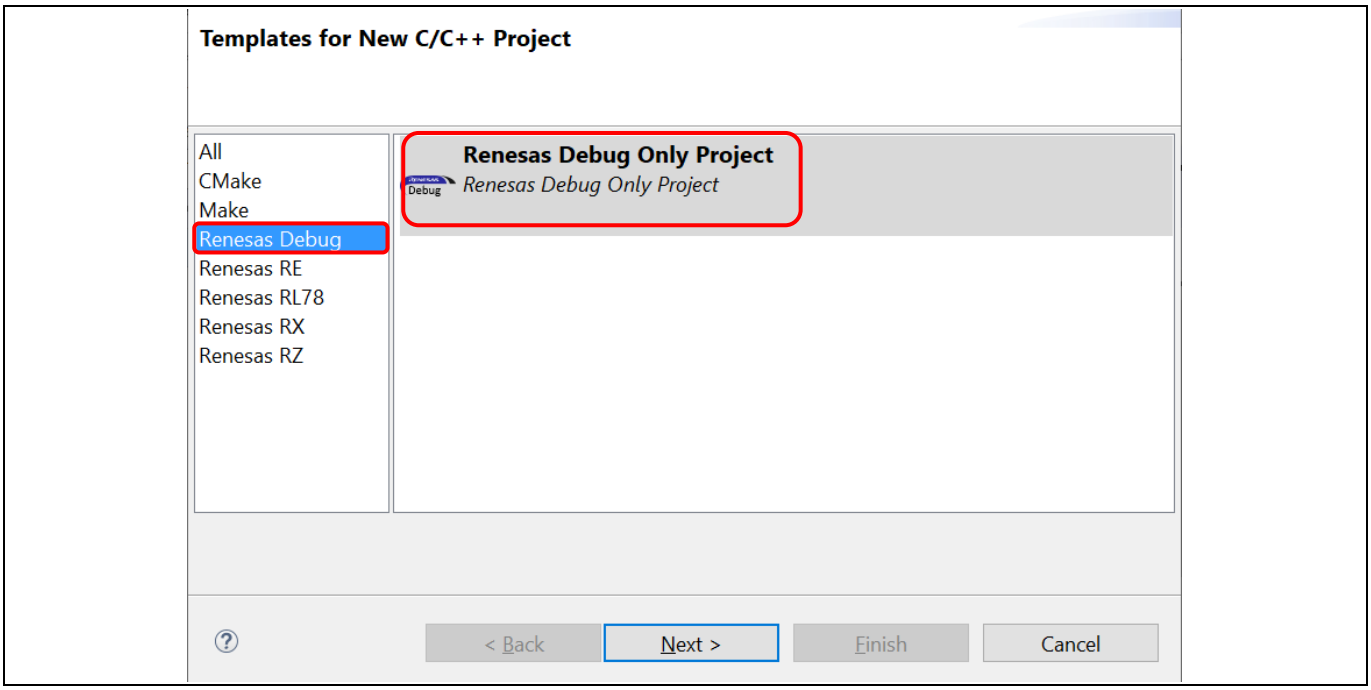


Figure 3-10 Specify the project template

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

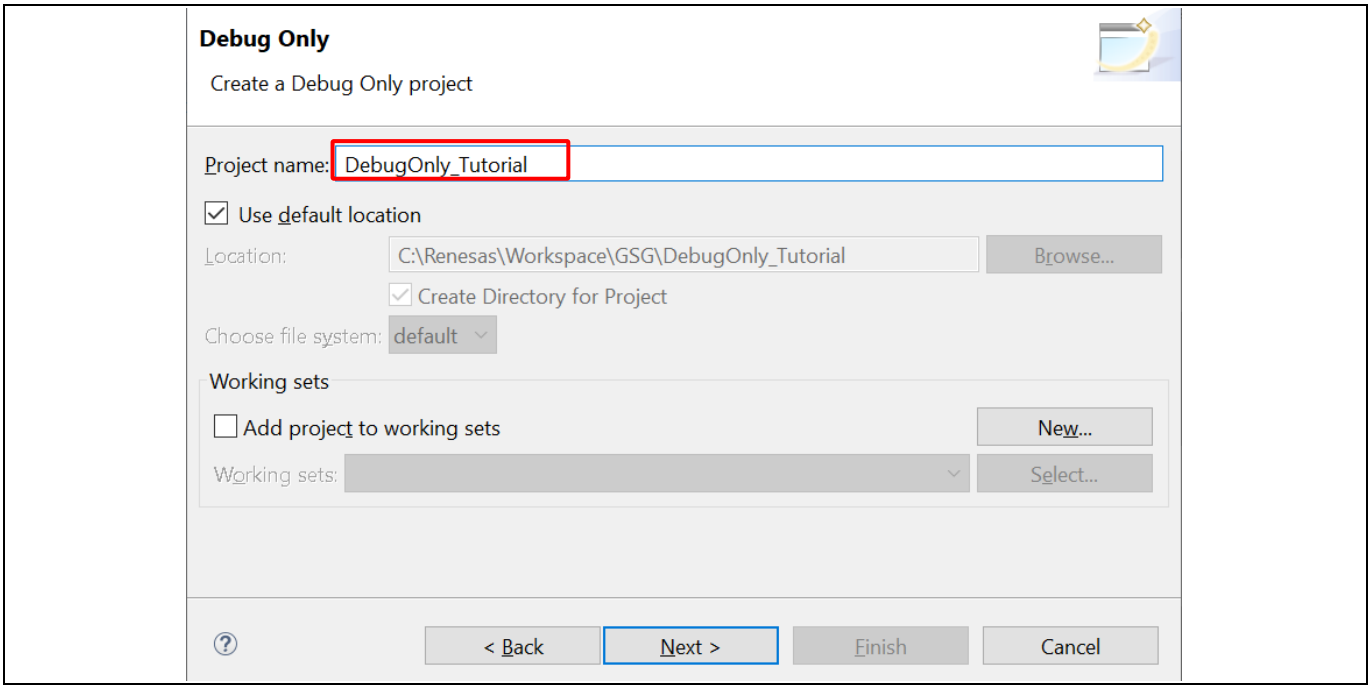


Figure 3-11 Specify project name

4. Select debug hardware (e.g. “E2 (RH850)”), target device (e.g. R7F701002xAFP). Note that these settings should be consistent with the settings to build the executable file. Then specify the location of prebuilt executable file (i.e. executable file built in other IDEs) which should be built as ELF/DWARF format to be recognized by debugger. Click [Finish] to create the project.

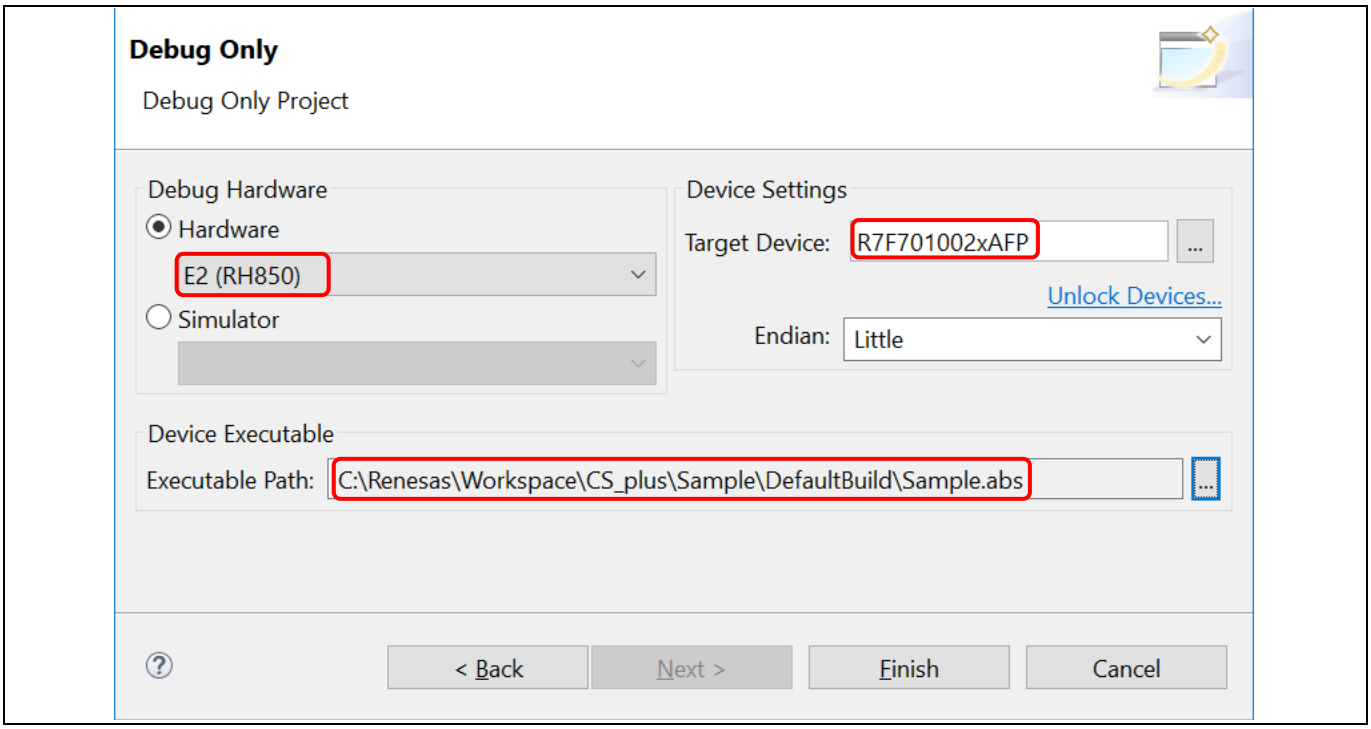


Figure 3-12 Specify project settings

5. The project named “DebugOnly_Tutorial” is created. User can only modify the debug configuration of this project and start the debugging.

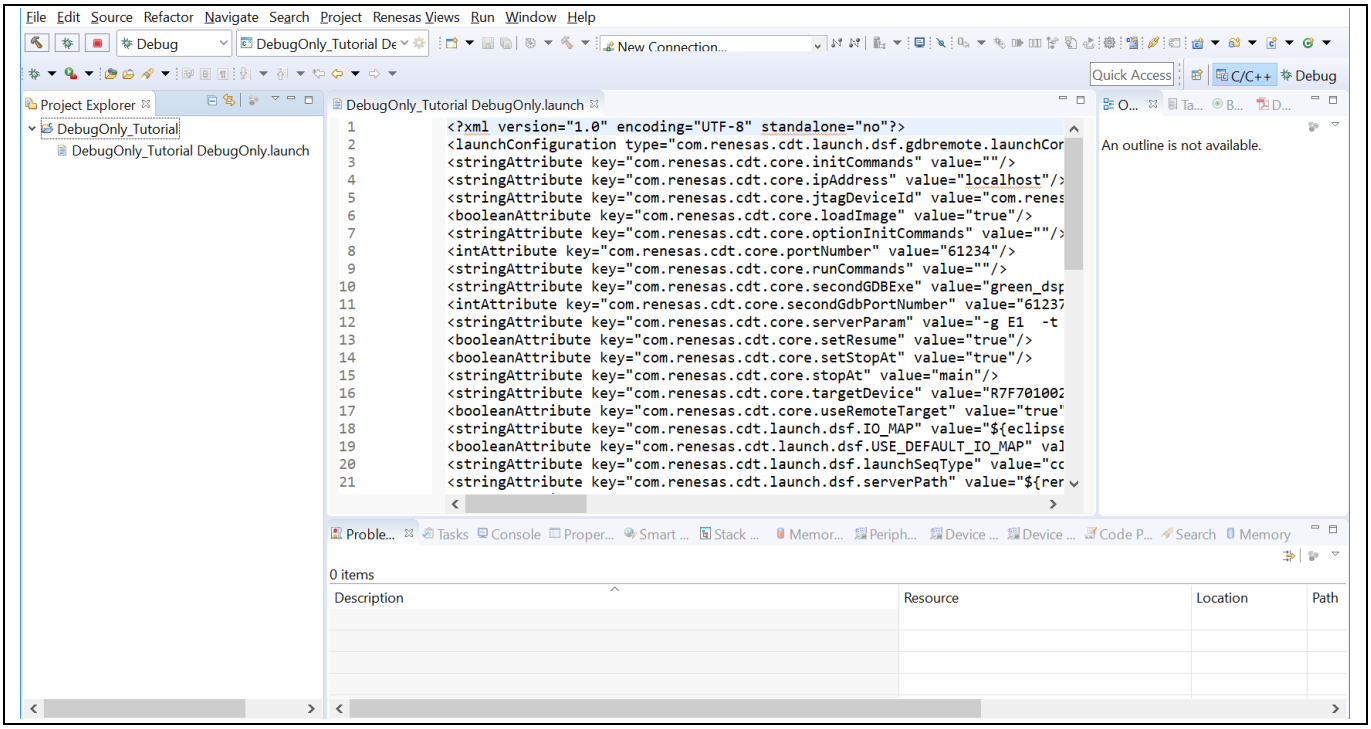


Figure 3-13 Debug only project is created

3.3 Import Existing Projects into Workspace

The migration guideline between integrated development environments (i.e. import CS+/HEW projects to e² studio, or export to CS+) can be found at the following site.

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

To import an existing e²s project to current workspace, please follow below instructions. These steps import a sample project from Renesas website to use for demonstrating debugging features in section 5.4.

1. Download the sample code for RX64M by searching for “RX64M Renesas Starter Kit Sample Code for e² studio” from Renesas website: https://www.renesas.com/search/keyword-search.html.

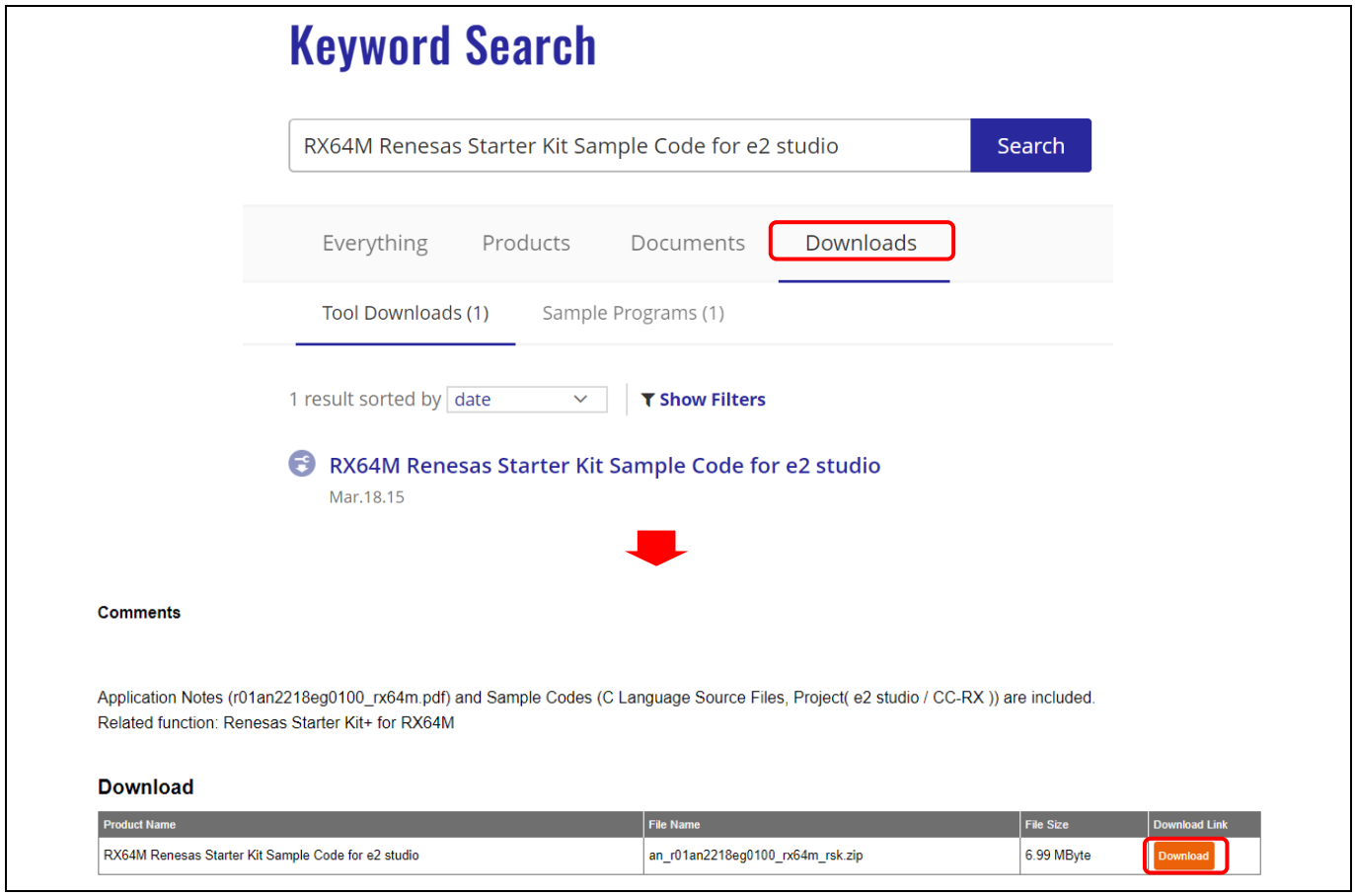


Figure 3-14 Download the Sample Code

2. After extract the package, find the project “Tutorial”

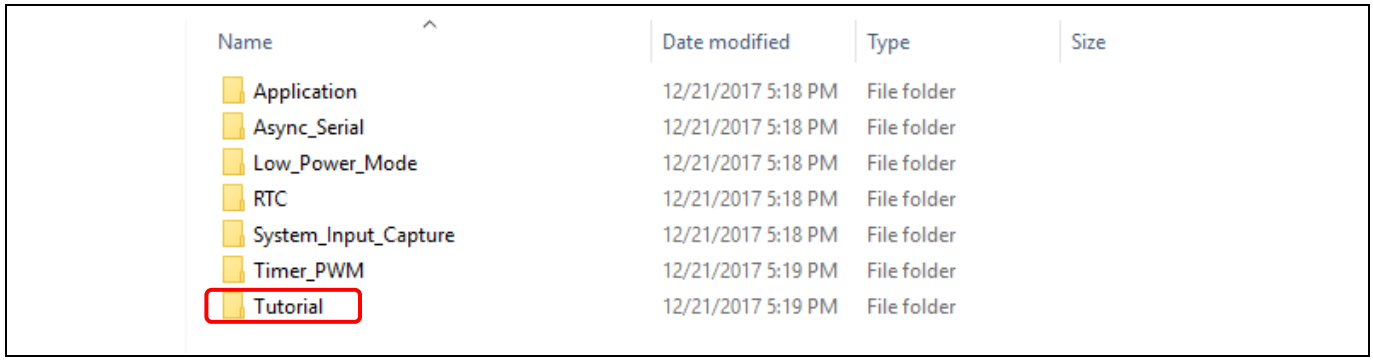


Figure 3-15 The Sample project

3. In e² studio, select [File] → [Import]

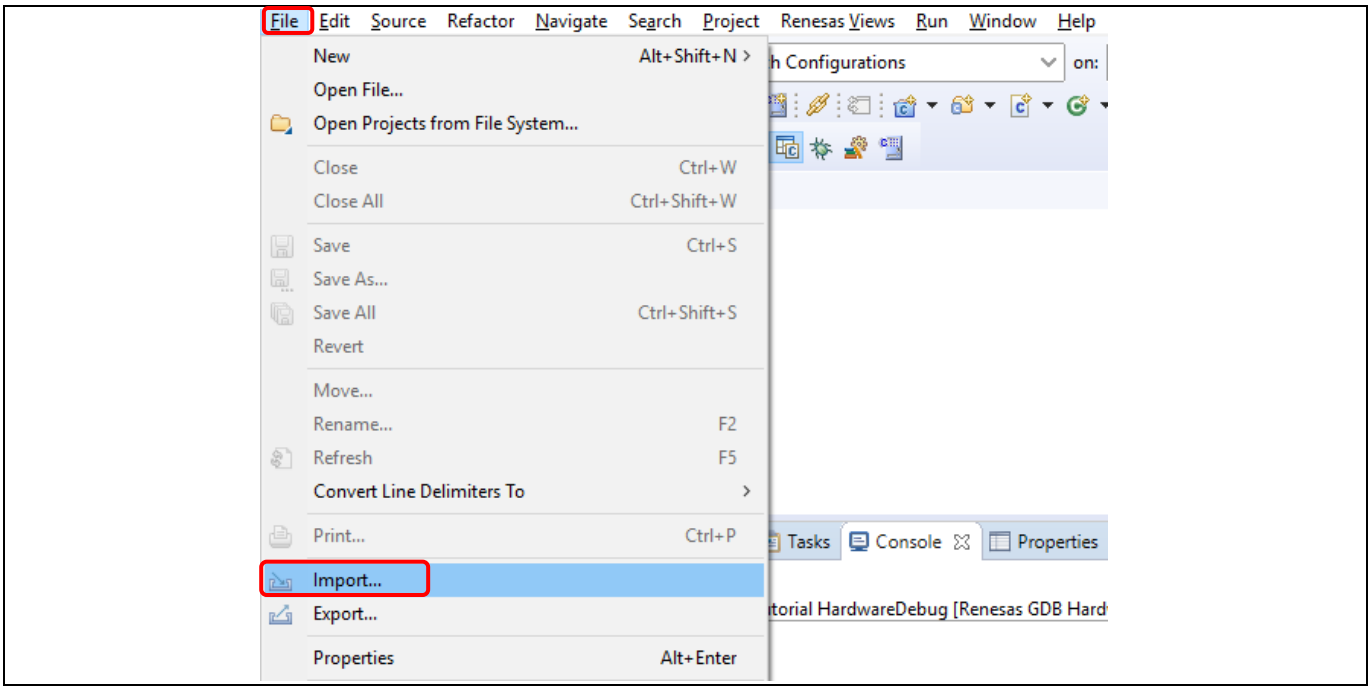


Figure 3-16 Import the sample project

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

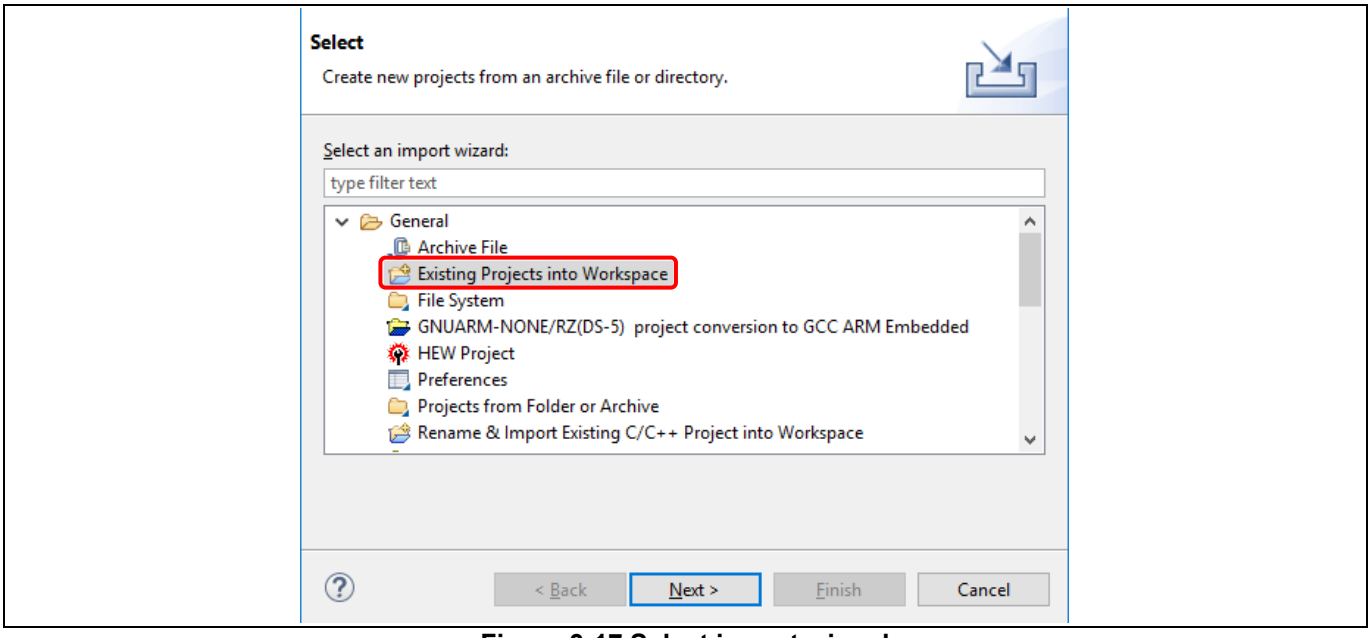


Figure 3-17 Select import wizard

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

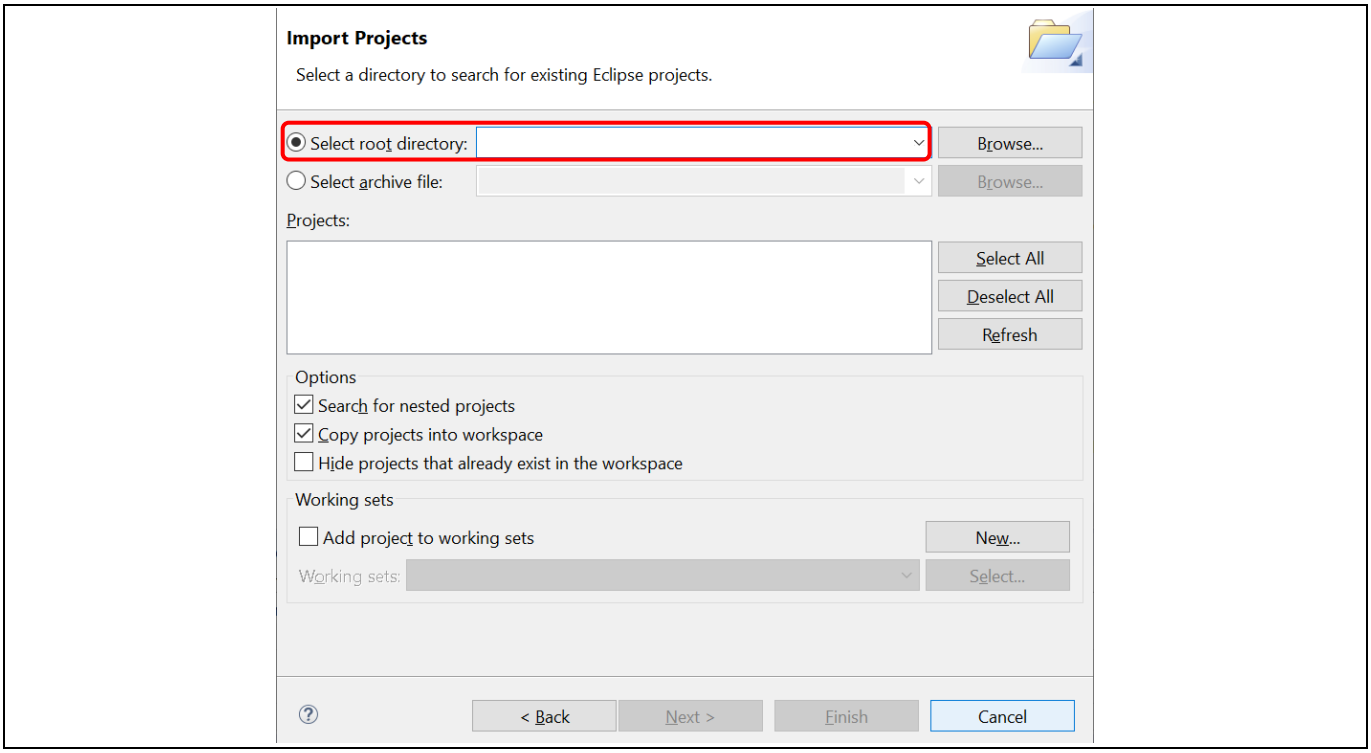


Figure 3-18 Select project location to import

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

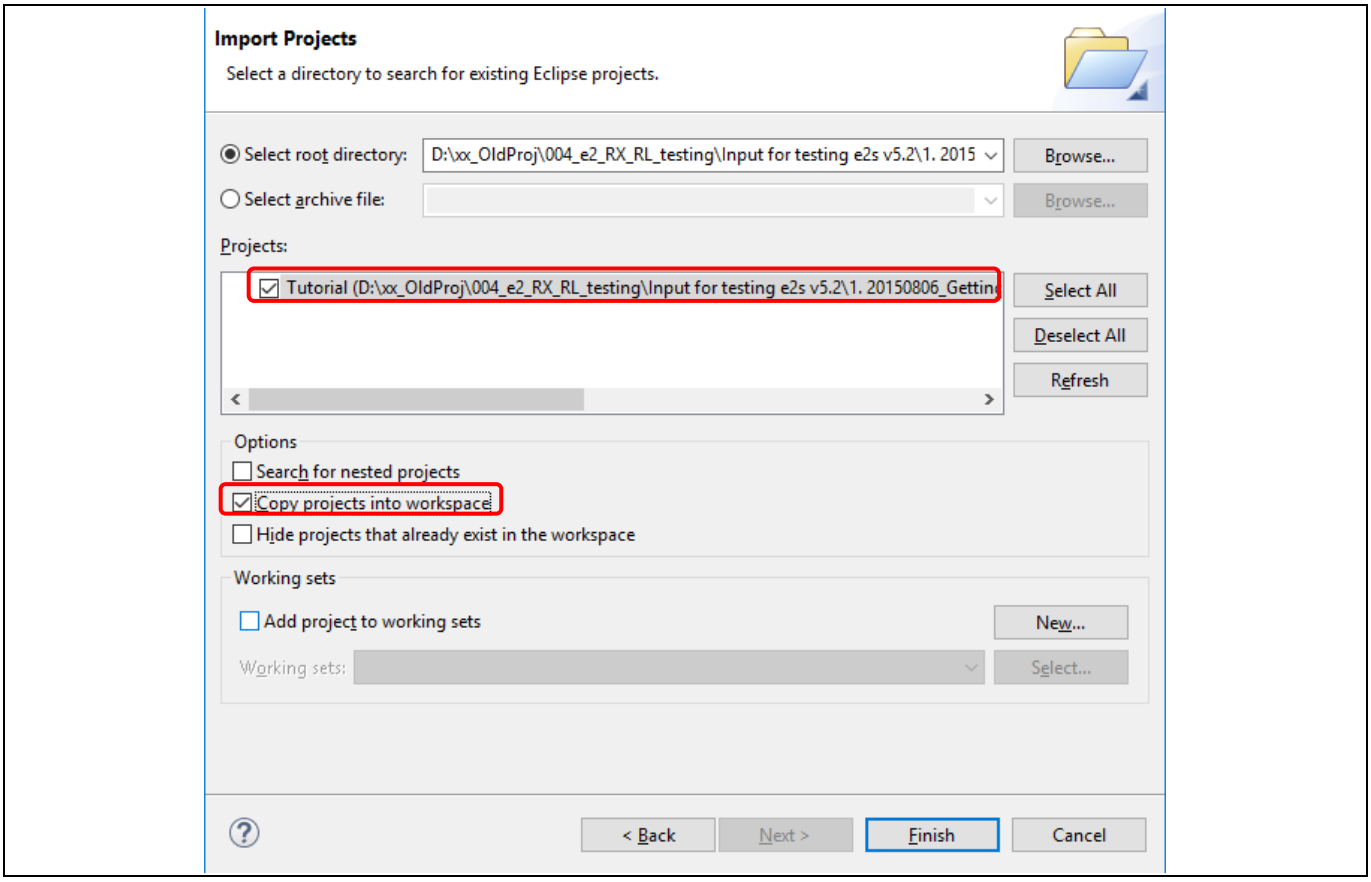


Figure 3-19 Complete project import

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

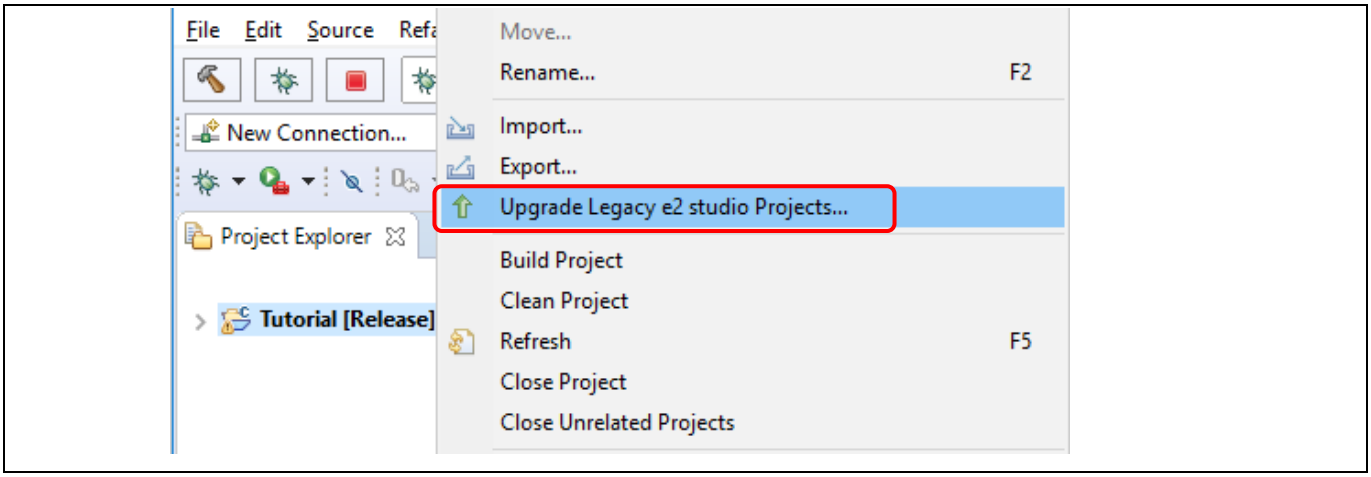


Figure 3-20 Upgrade the imported project

8. Select “Tutorial” project and click [Finish]

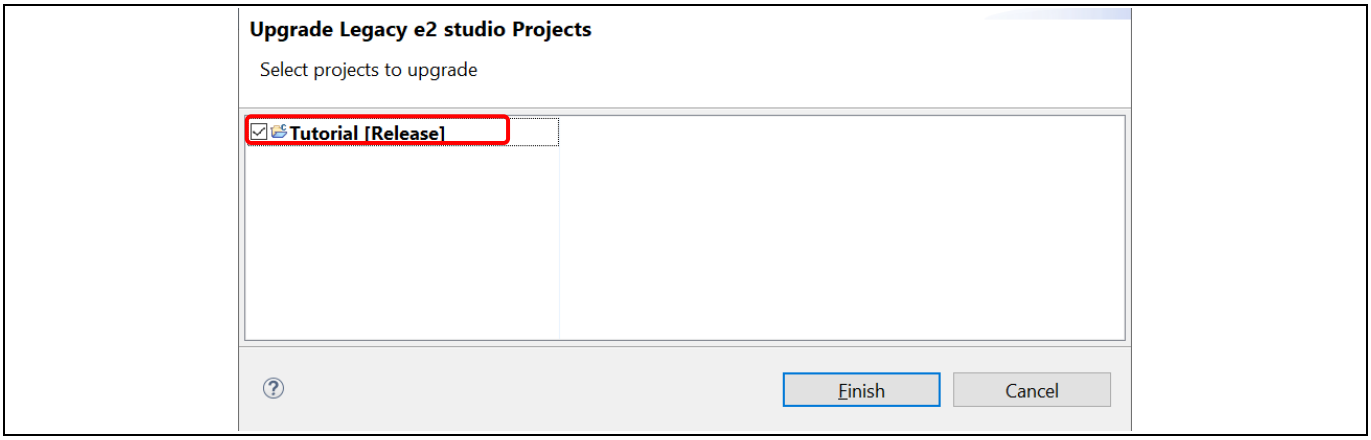


Figure 3-21 Finish the upgrading

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].

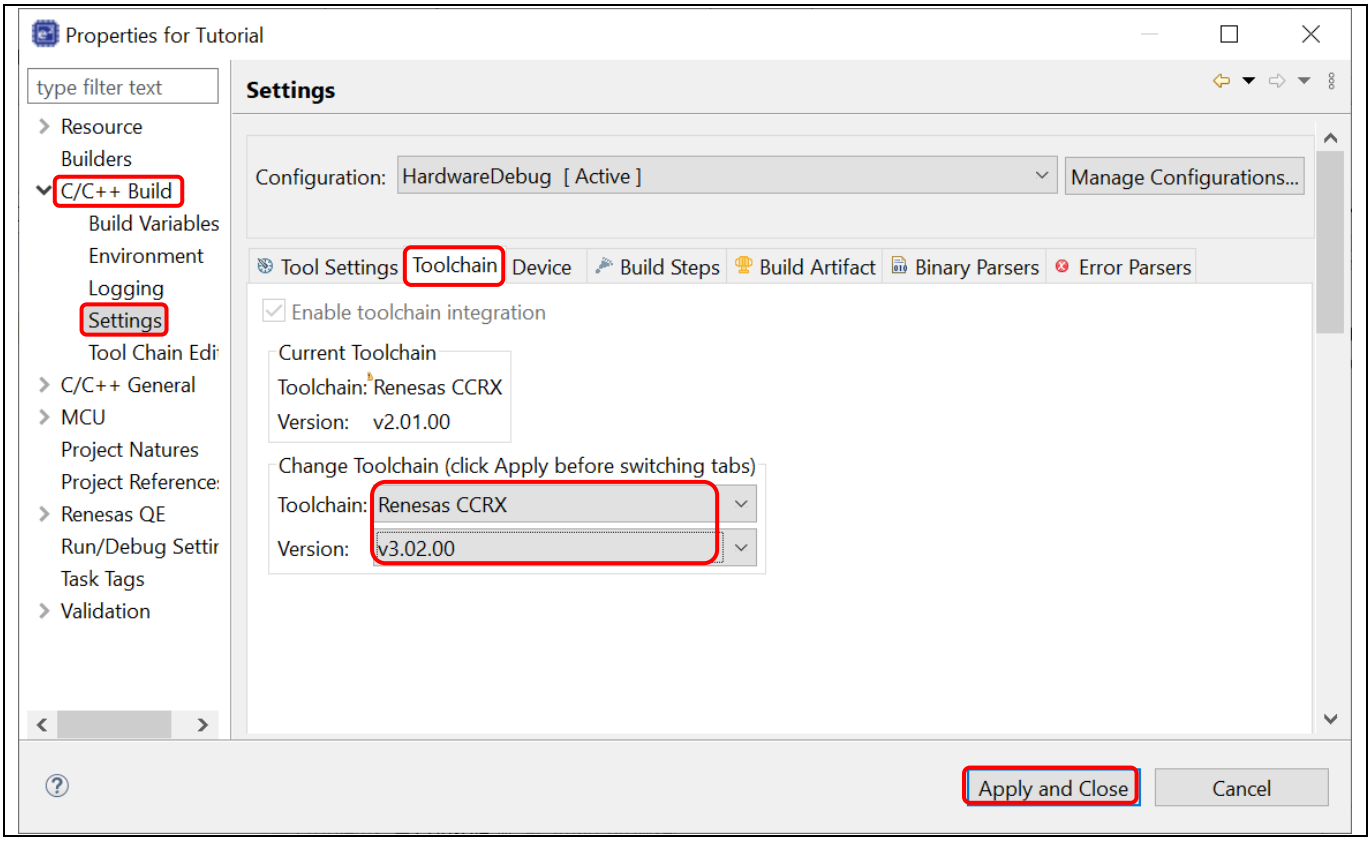


Figure 3-22 Update project toolchain

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

4. Build

This chapter describes the build configurations and key build features for e² studio IDE.

4.1 Build Option Settings

A new project built with default option can work properly. However, if user would like to change build options (e.g. toolchain version, optimization options, etc.), please follow the following steps before building the project.

1. Right click on project “Tutorial” and select [Properties] or click button 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.

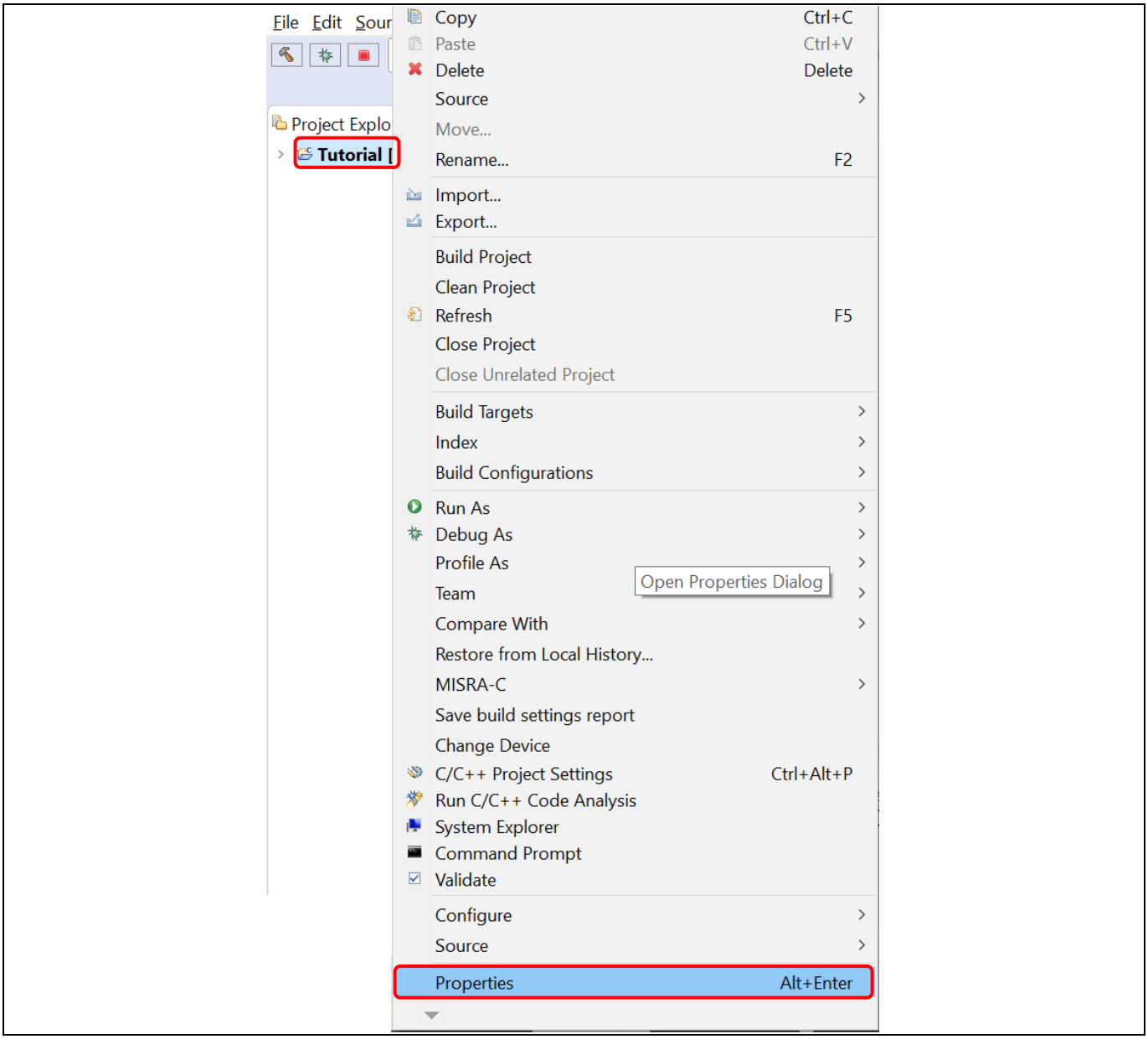


Figure 4-1 Open the Properties window

2. Click [C/C++ Build] → [Settings] → [Toolchain] to view or change toolchain version.

   Click the “Versions” option to change toolchain version (if additional toolchain is installed).

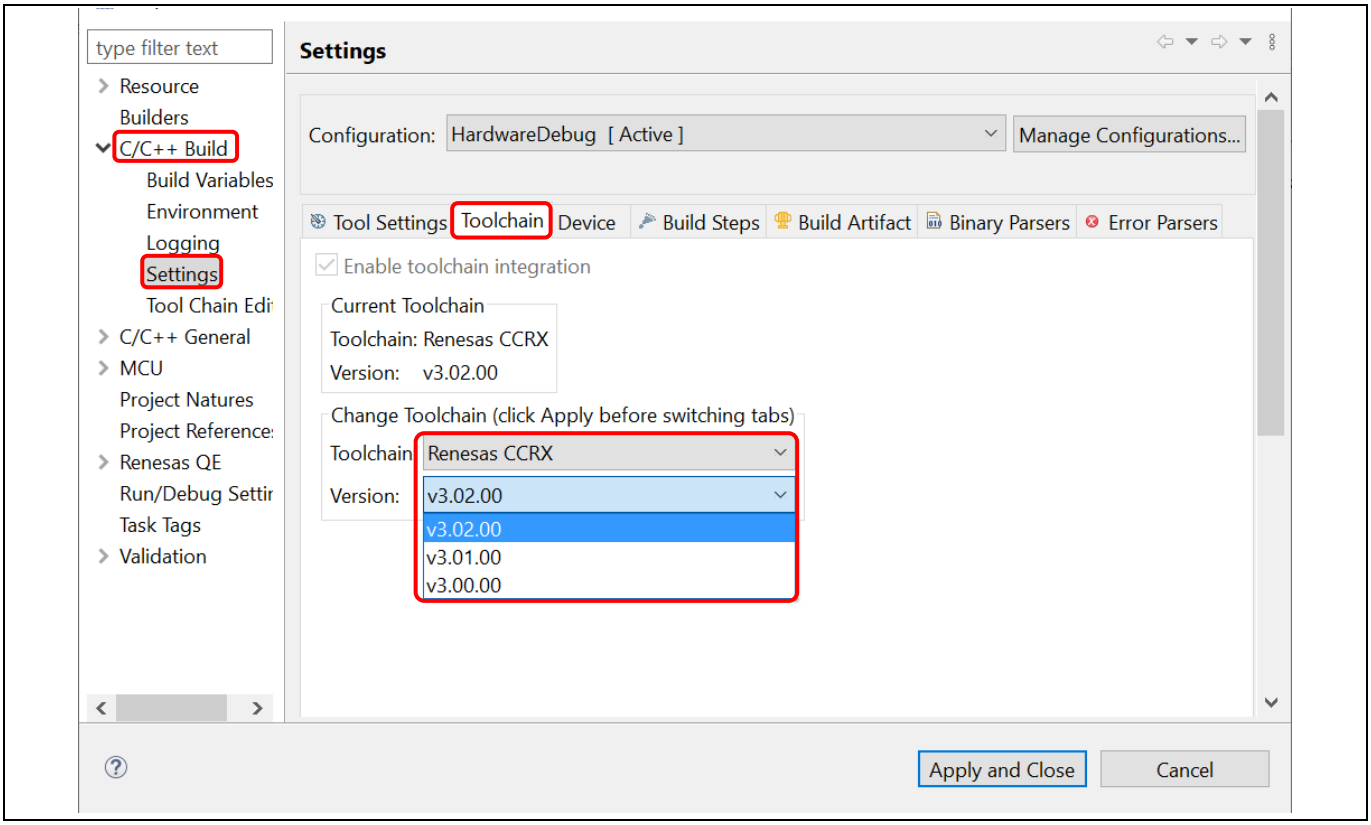


Figure 4-2 Change Toolchain Version

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

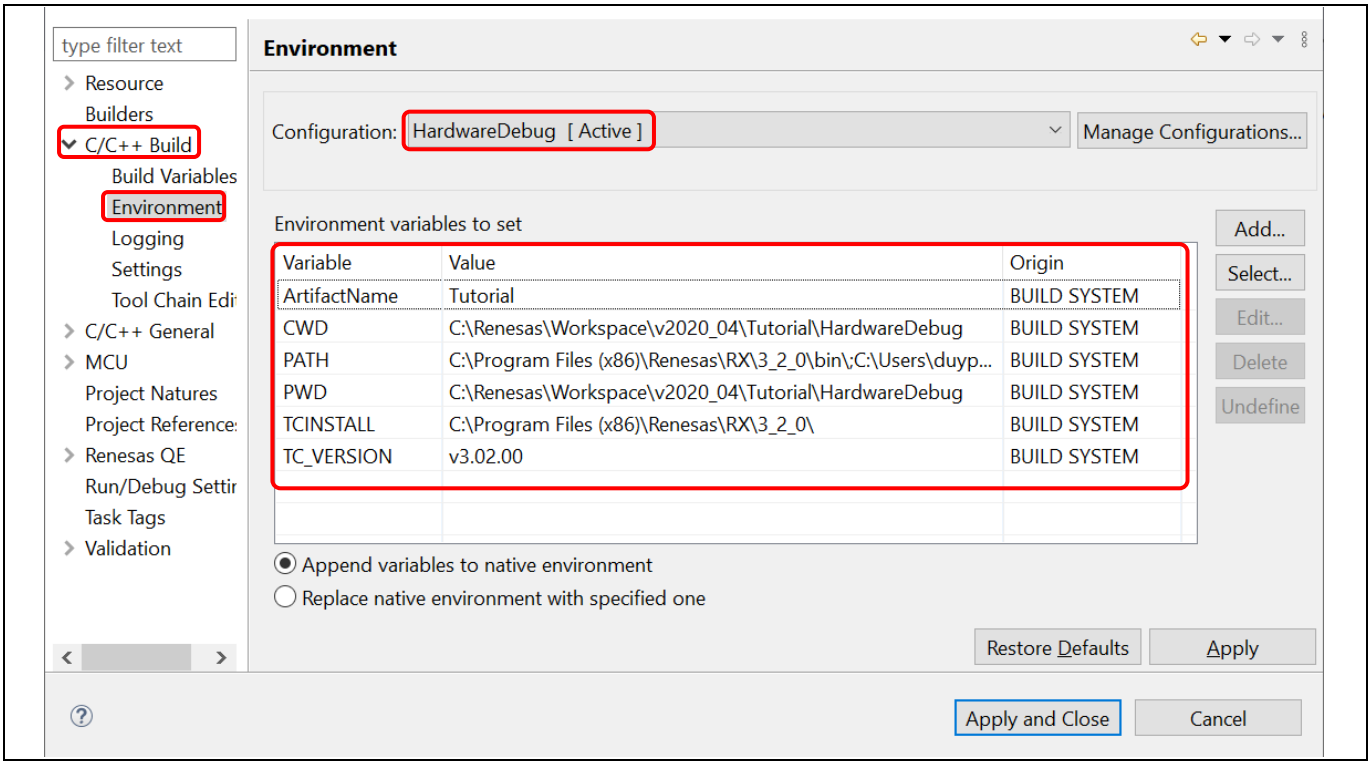


Figure 4-3 Build Environment Settings

4. Setting Build Options

Right click on a project in the Project Explorer and select [Properties] or click button to open the Properties window.

Build options for compiler and linker, etc. can be set on "C/C ++ Build" → "Settings"→ “Tool Setting” tab.

User could set all build settings under ‘Tool Settings’ tab.

The “Build configuration” can be switched via “Configuration:” dropdown list at the top of the window. Each build configuration manages a set of build options.

Click [Apply and Close] to save the build setting changes.

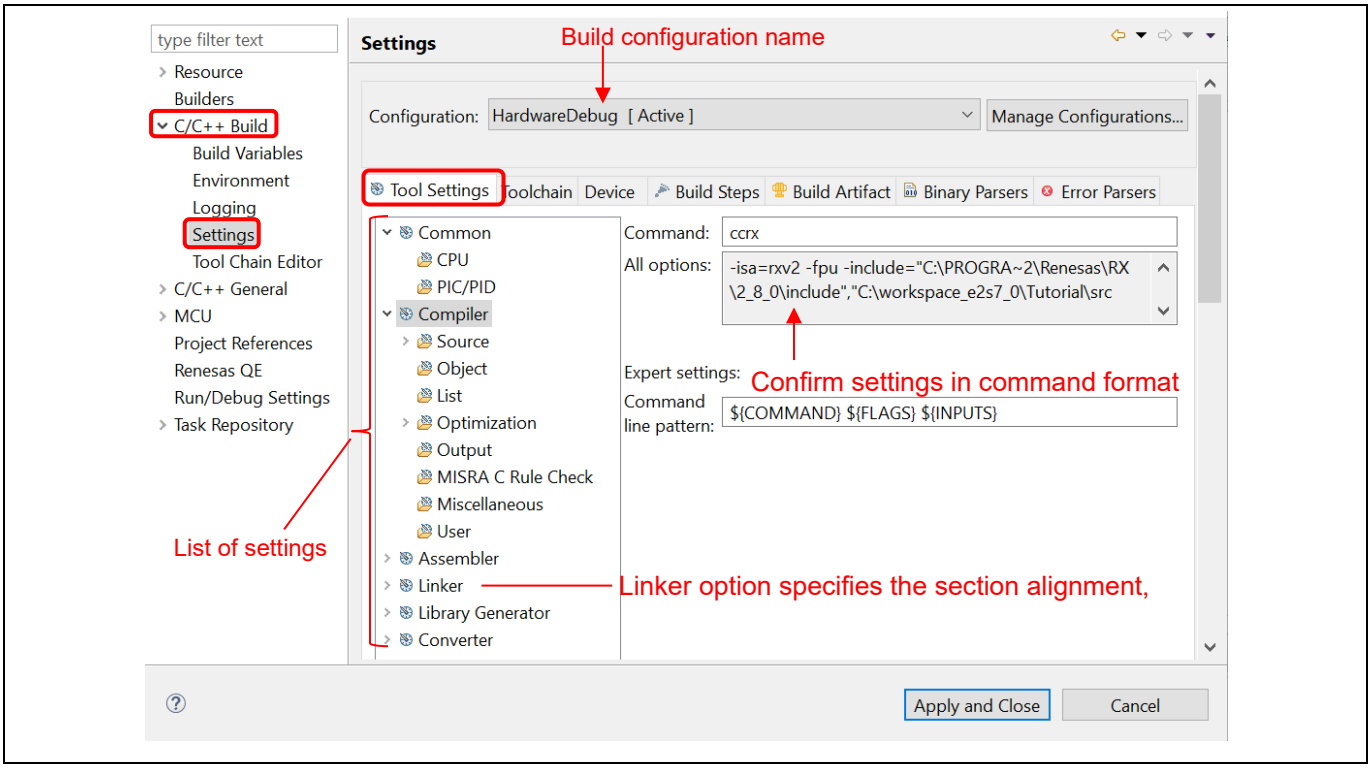


Figure 4-4 Build Option Settings

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 (x86)\Renesas\RX\3_2_0\doc\`”.

Note: There is "*Toolchain Editor*" under "C/C++ Build", **please do not change the configuration**. The Toolchain editor is used for toolchains which are NOT supported by Renesas build support plugins.

4.2 Build A Sample Project

A project can be built by the steps below:

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

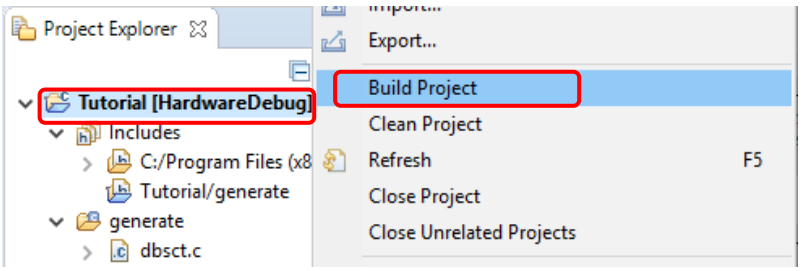


Figure 4-5 Build a Sample "Tutorial" Project

2. Check 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 e² studio IDE.

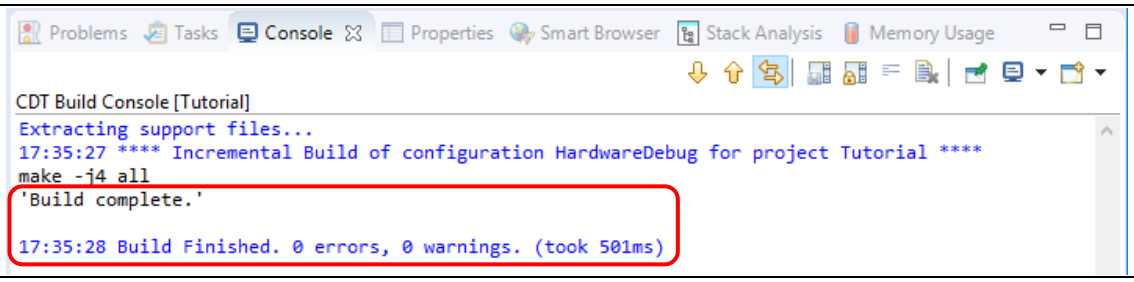


Figure 4-6 Project is built successfully.

3. In some cases, the build can be unsuccessful. The console window will show error messages, please check it and revise the source code or configuration and rebuild the project

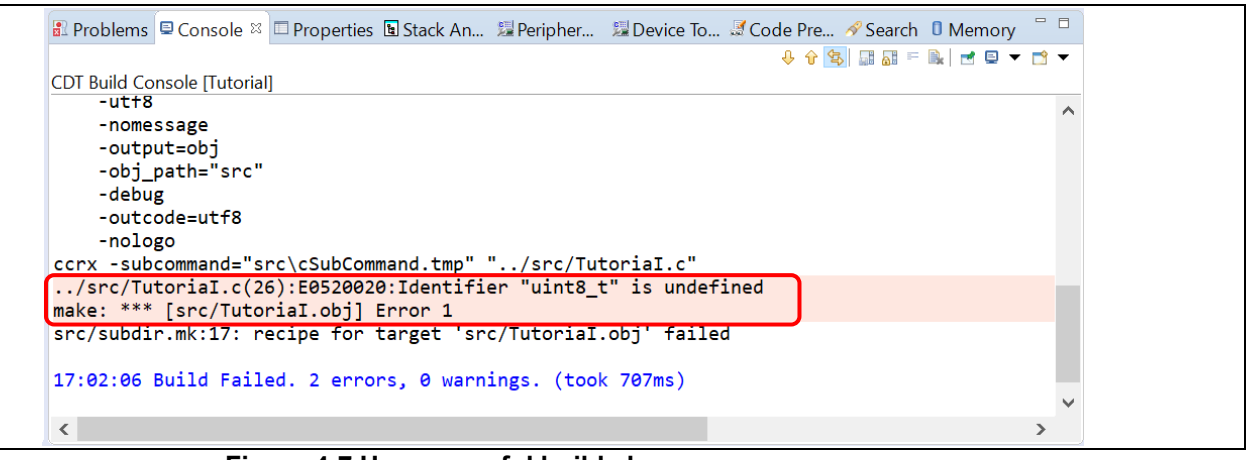


Figure 4-7 Unsuccessful build shows error messages

4.3 Export Build Configuration Settings

The Project Reporter feature can export project and build configuration settings from e² studio IDE to a file for easy checking and comparison of project/build environment settings.

1. Right-click at [Project Explorer] to pop up the context menu
2. Select [Save build settings report] to save build settings report

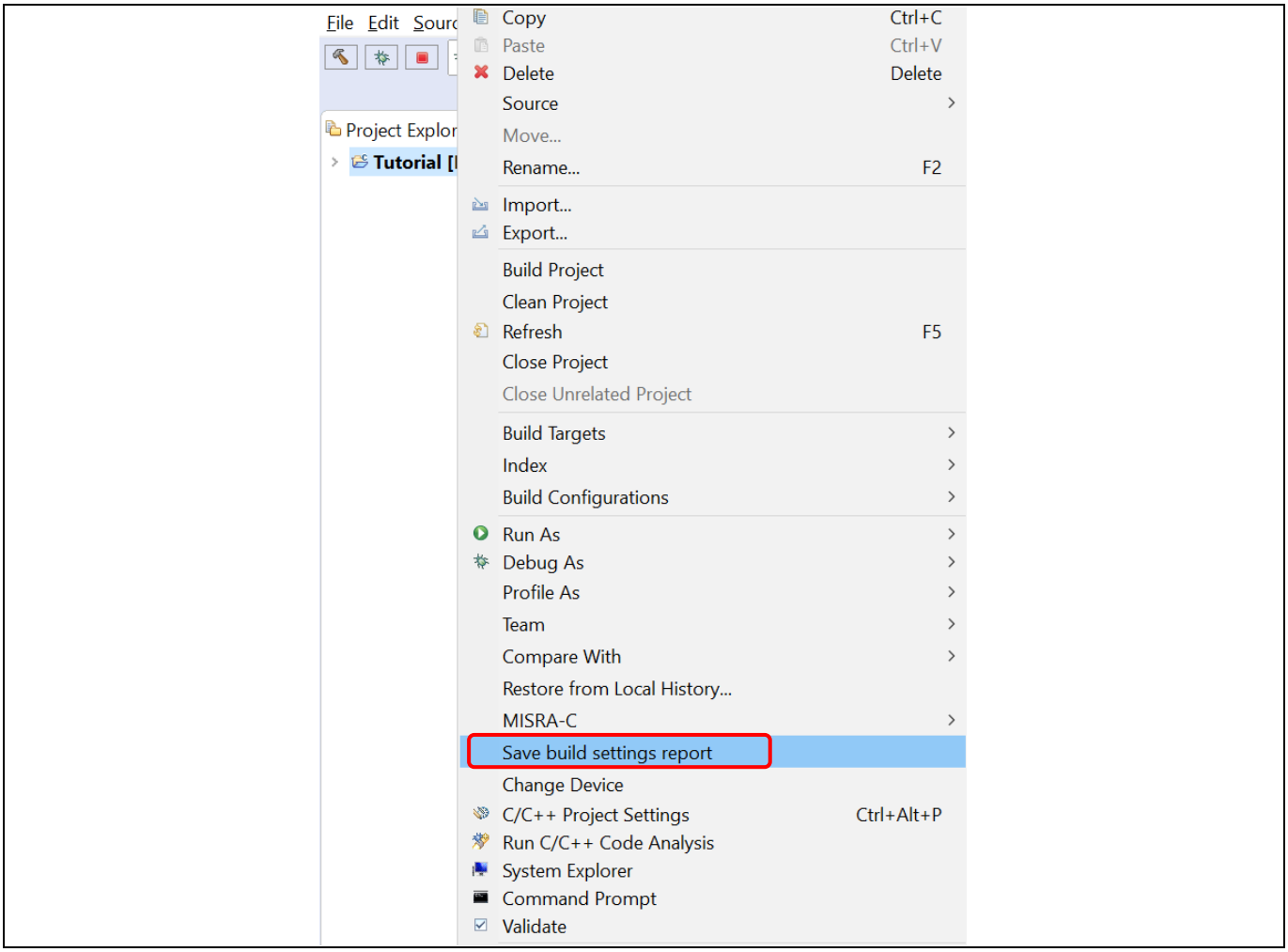


Figure 4-8 Project Reporter

5. Debug

This chapter describes the usage of debug configuration and key debugging features for e² studio. 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.

Firstly, open “Tutorial” project workspace in e² studio IDE and click [Debug] perspective.

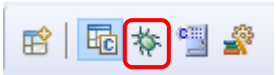


Figure 5-1 Switch to [Debug] Perspective

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 “e² 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.

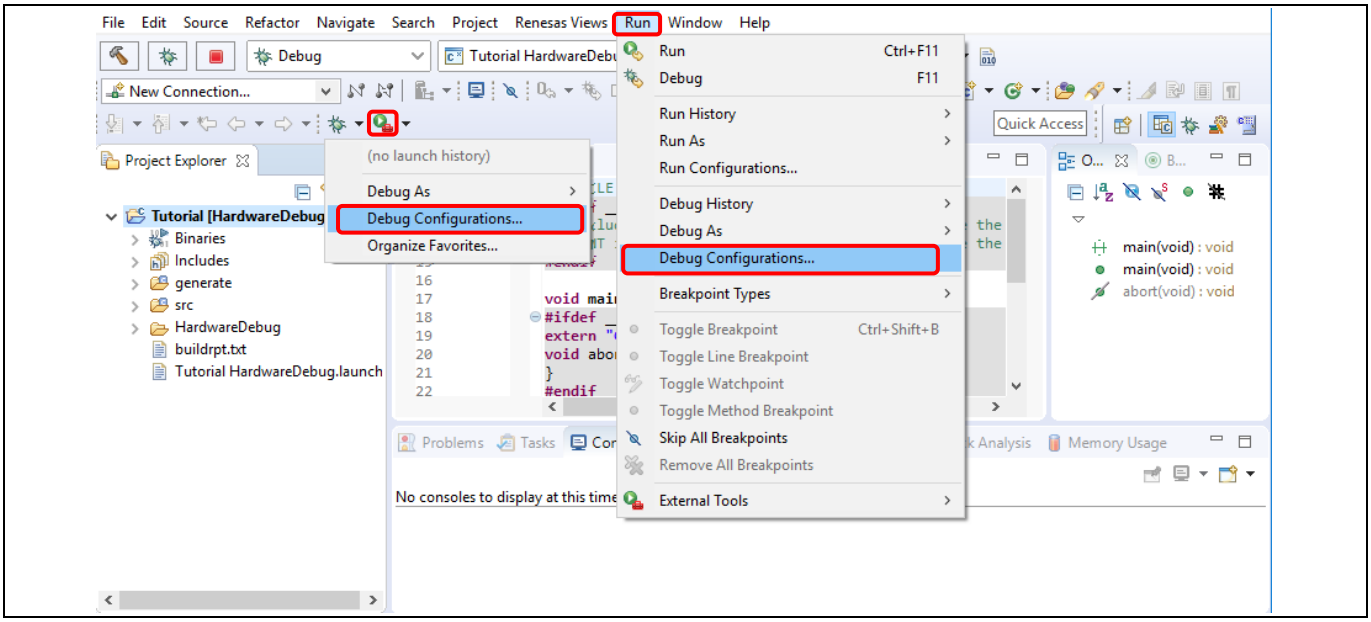


Figure 5-2 Open Debug Configurations Window

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”.

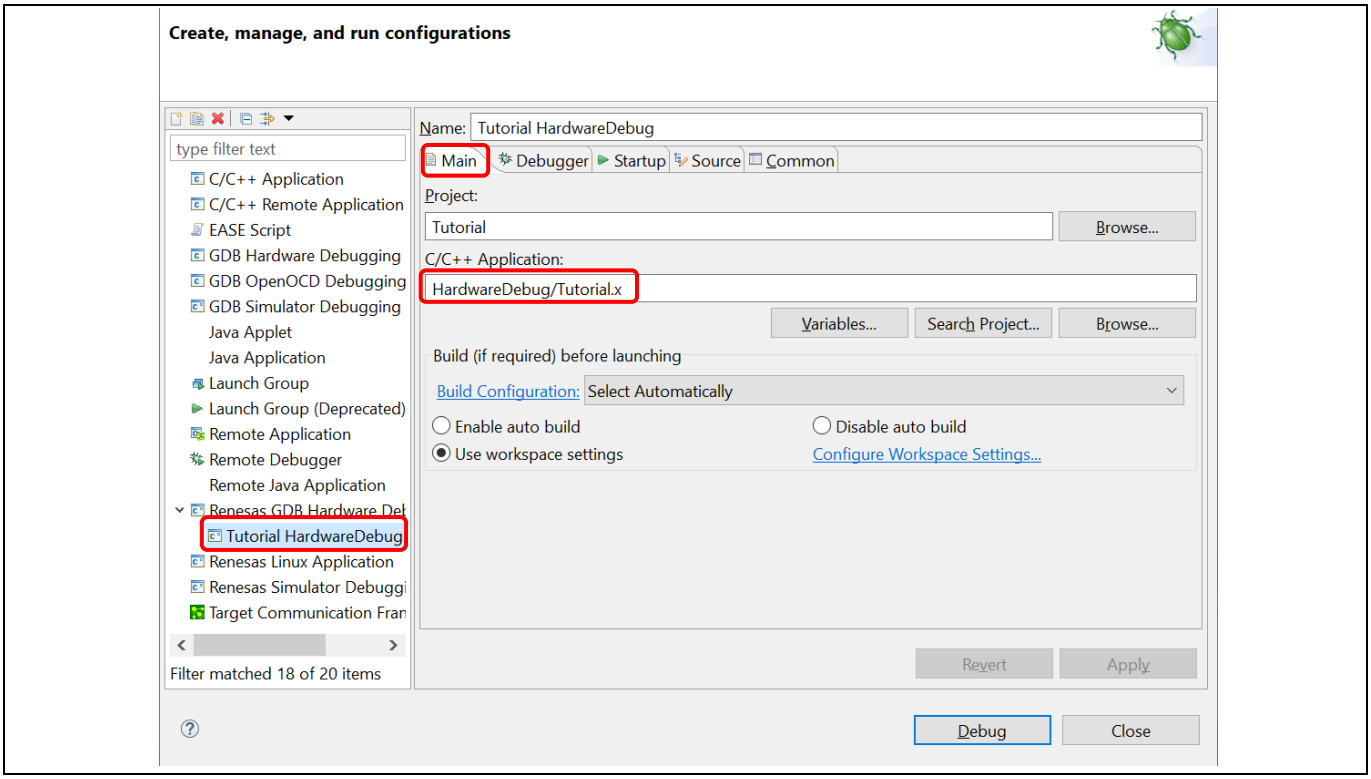


Figure 5-3 Select Load Module

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

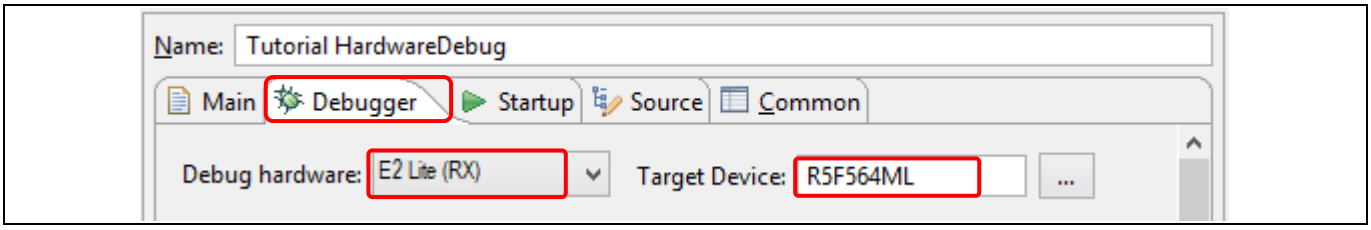


Figure 5-4 Select Target Device

4. Under the [Debugger] tab, go to the [Connection Settings] sub tab which is related to emulator connection. The following example is based on the environment with 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”
  - Hot plug = “No”

  Hot plugin feature is only available with the device which has the capability. Please refer to device Hardware manual for "On chip debugger" specifications for the details.

- Power
  - Power Target From The Emulator (MAX 200mA) = “No”

Choose "Yes" if you would like to supply power through an emulator, when external power is unplugged. Choose "No" if external power is plugged.

- Communication Mode
  - Mode = “Debug Mode”

  Another communication mode "Write On Chip Flash Memory" is used for flashing codes including ID code area, however debugger will be disconnected after flash.

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.

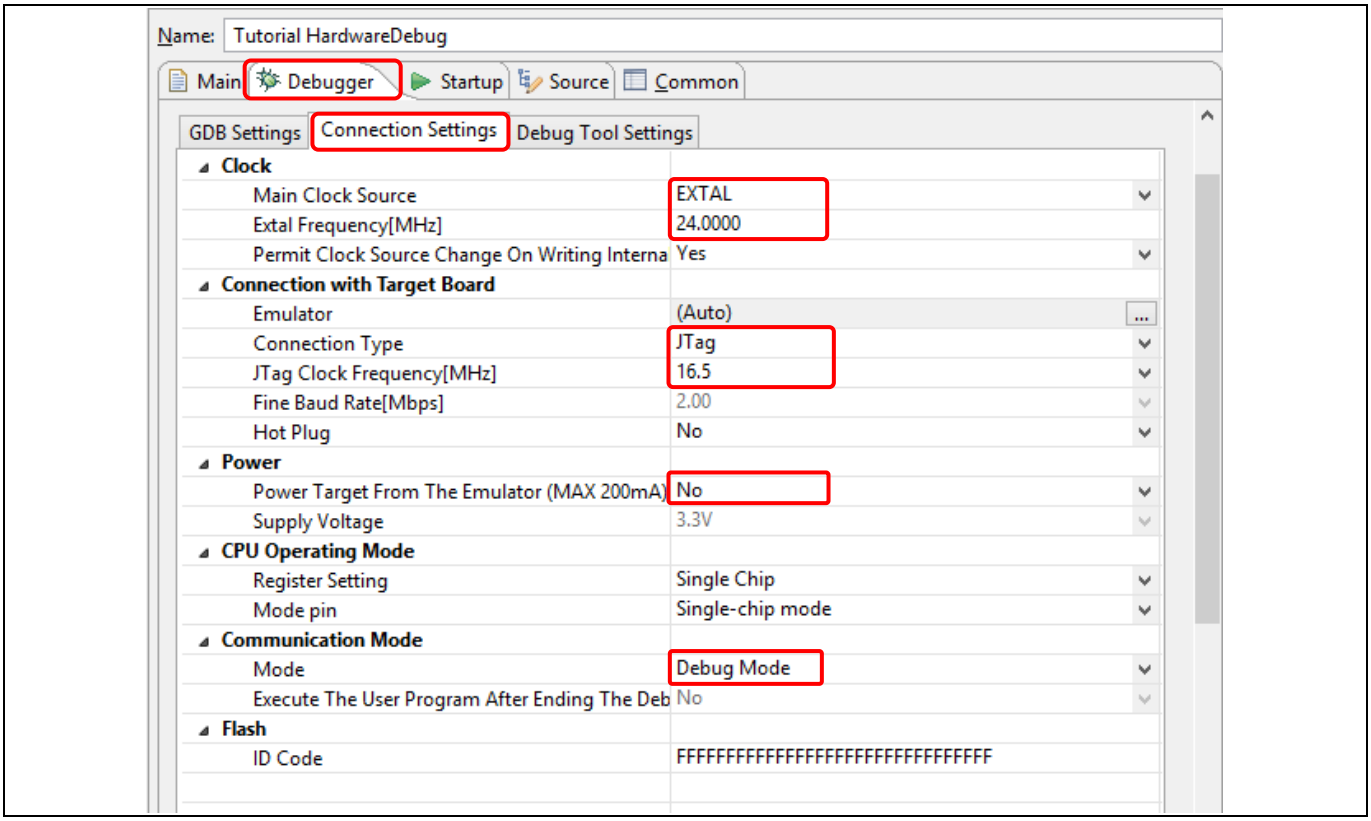


Figure 5-5 Change Connection Setting

5. Switch to [Debug Tool Settings] sub tab which is related to debugger behavior, please refer to the e² studio Help content at "e² studio User Guide" → "Debugging Projects" for the details.

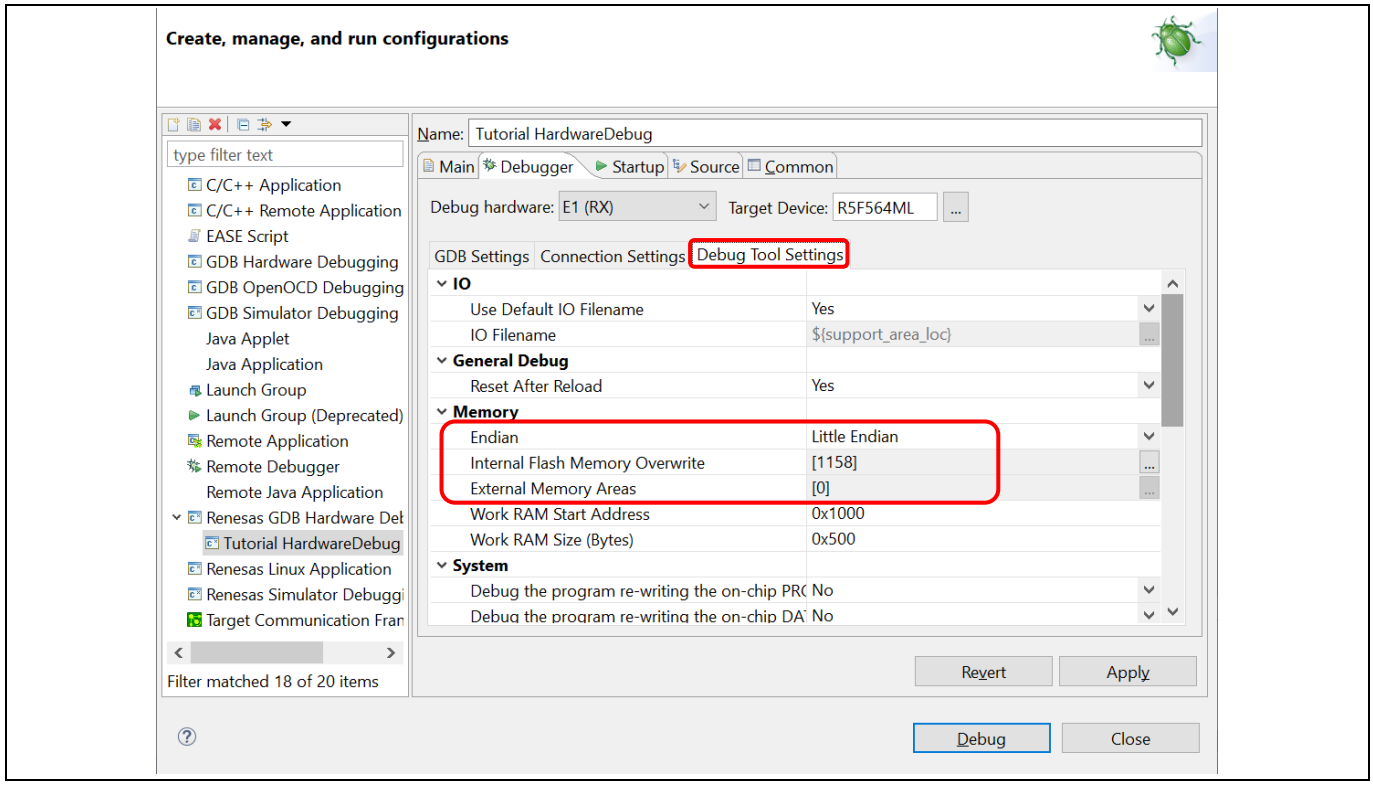


Figure 5-6 Change Debug Tool Settings

- Memory
  - Endian = “Little Endian”

    Endian setting of debugger memory reference. This configuration does not affect to target program behavior.

  - “Internal Flash Memory Overwrite”, "External Memory Areas"

    These configuration control to allow/deny flashing blocks upon module downloads. Uncheck specific memory blocks if you would like to reserve the contents.

6. Click [Apply] button to confirm the settings. Then click [Debug] to launch debugger.

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”.

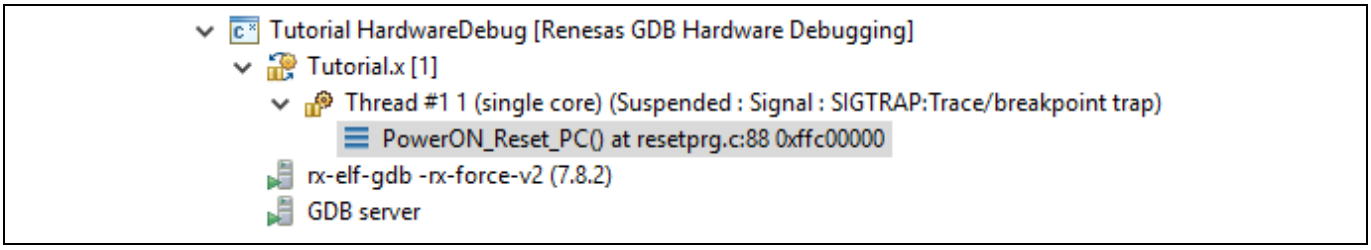


Figure 5-7 User Target Connection in the [Debug] View

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.

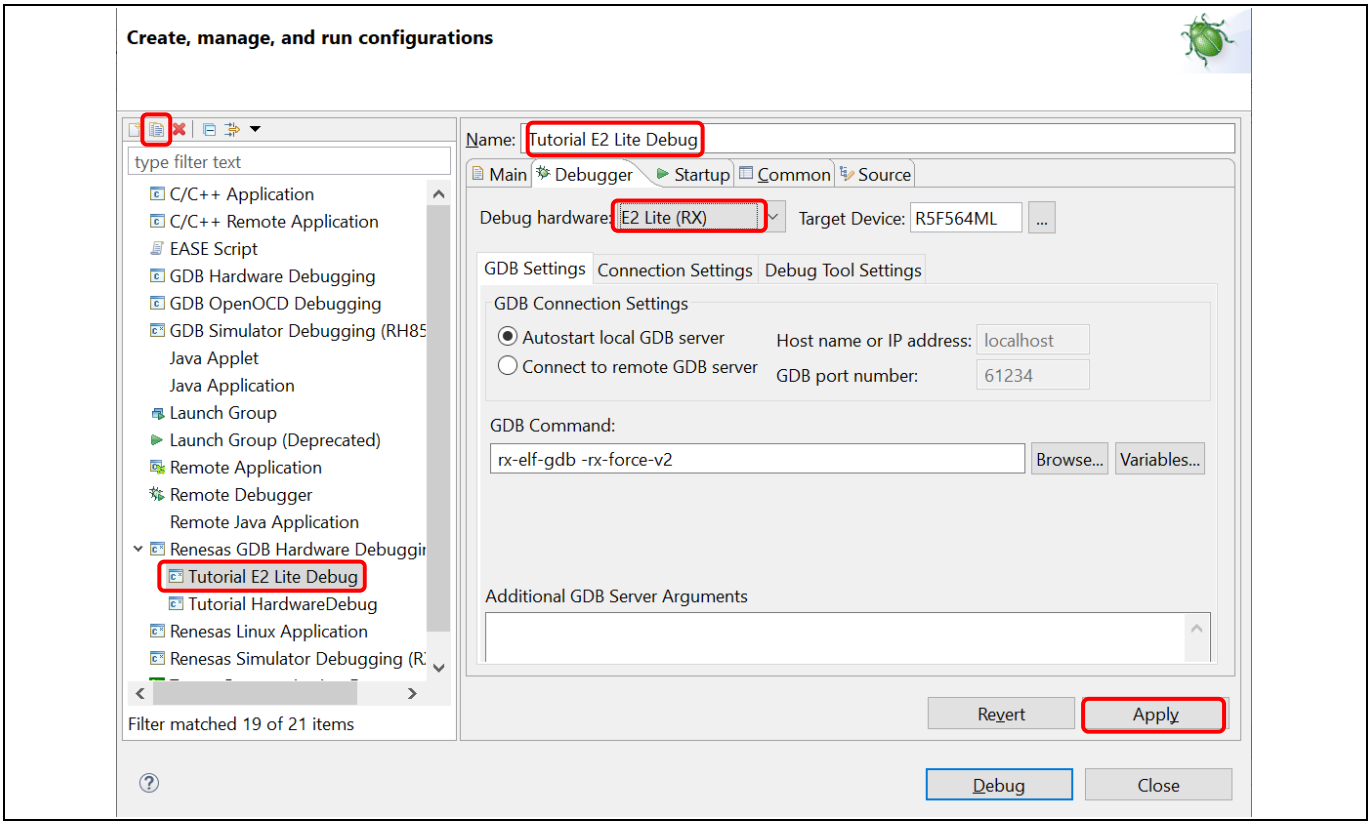


Figure 5-8 Duplicate A Selected Debug Launch Configuration

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

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

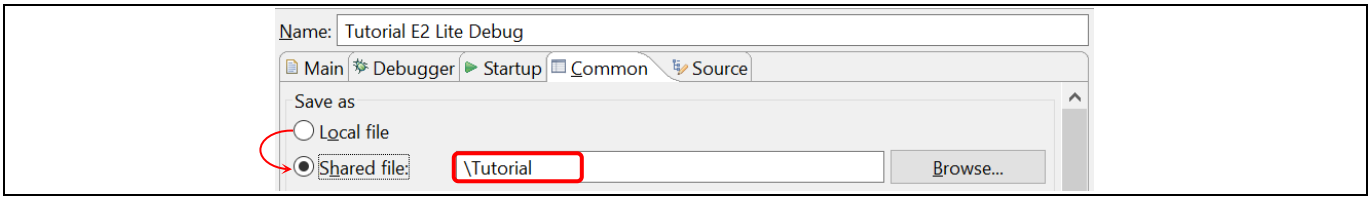


Figure 5-9 Attach Launch Configuration to Specific Project

5.3 Launch Bar

This section explains the usage of 'Launch Bar', which is supported from V6.0.0 or later version.
Launch Bar located in the toolbar area of e2 studio main window.

The interface is as shown below to build and debug for the selected launch target.

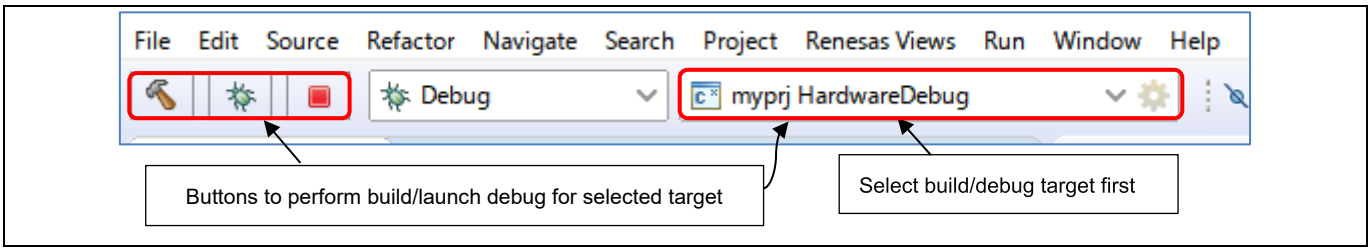


Figure 5-10 Launch Bar interface

Launch Bar buttons behave as follows:

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

  Note: There is another build button in the "File toolbar" that 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]

5.4 Basic Debugging Features

This section explains the typical Debug views supported in e2 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:

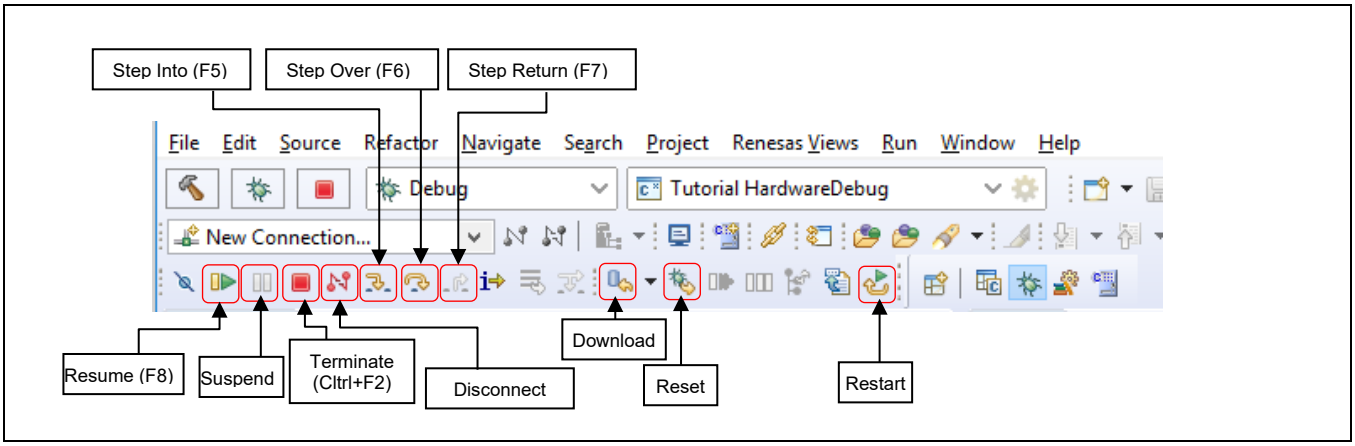


Figure 5-11 Useful Toolbars in Debug Views

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

The program can be paused by breakpoint or by 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 instruction in section 3.3.

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. e² 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:

- Right-click on the marker bar to pop up the context menu. For a hardware breakpoint, select [Breakpoint Types] → [e² 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.

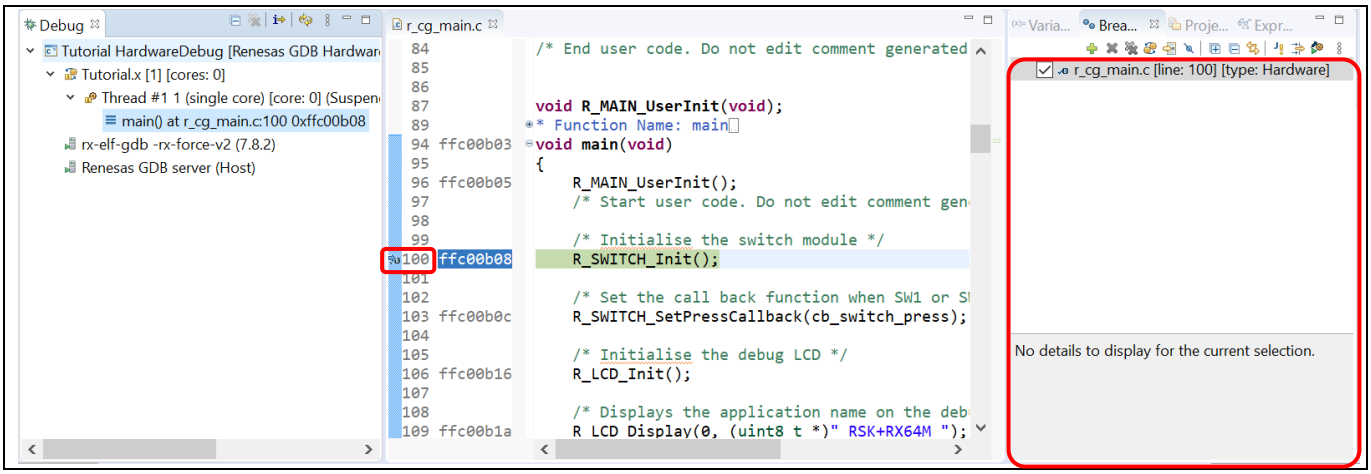


Figure 5-12 [Breakpoints] view

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.

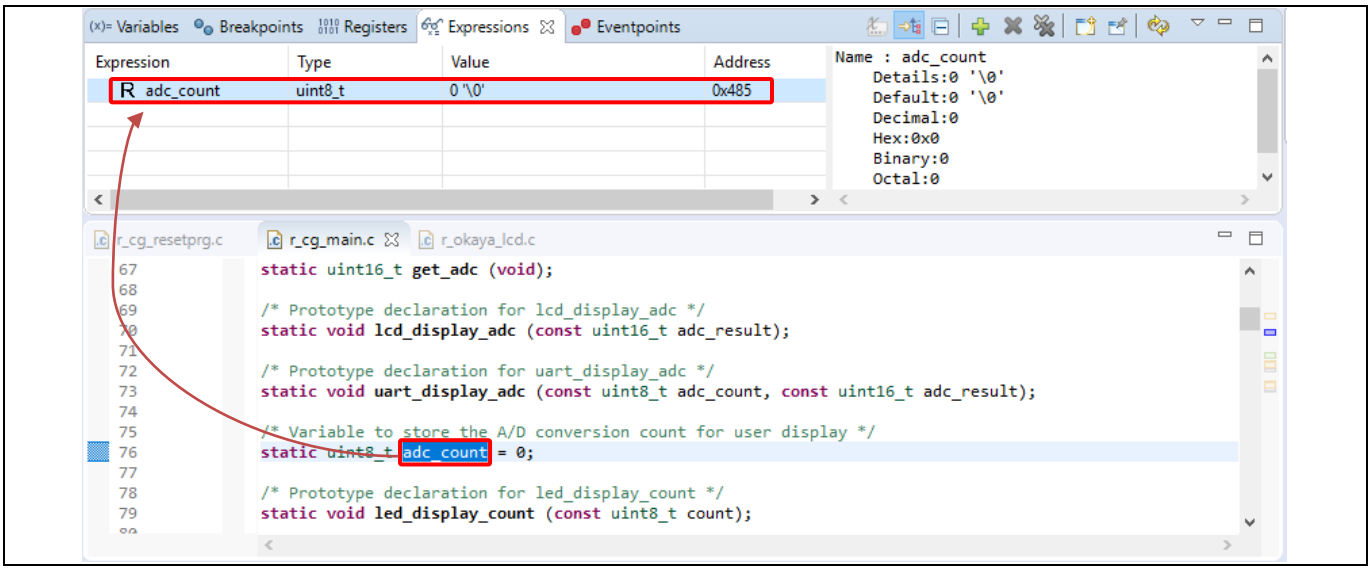


Figure 5-13 [Expressions] View

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.

Local variables can be added with the same way. However, the watch is not available when the program is running out of the scope of the variable.

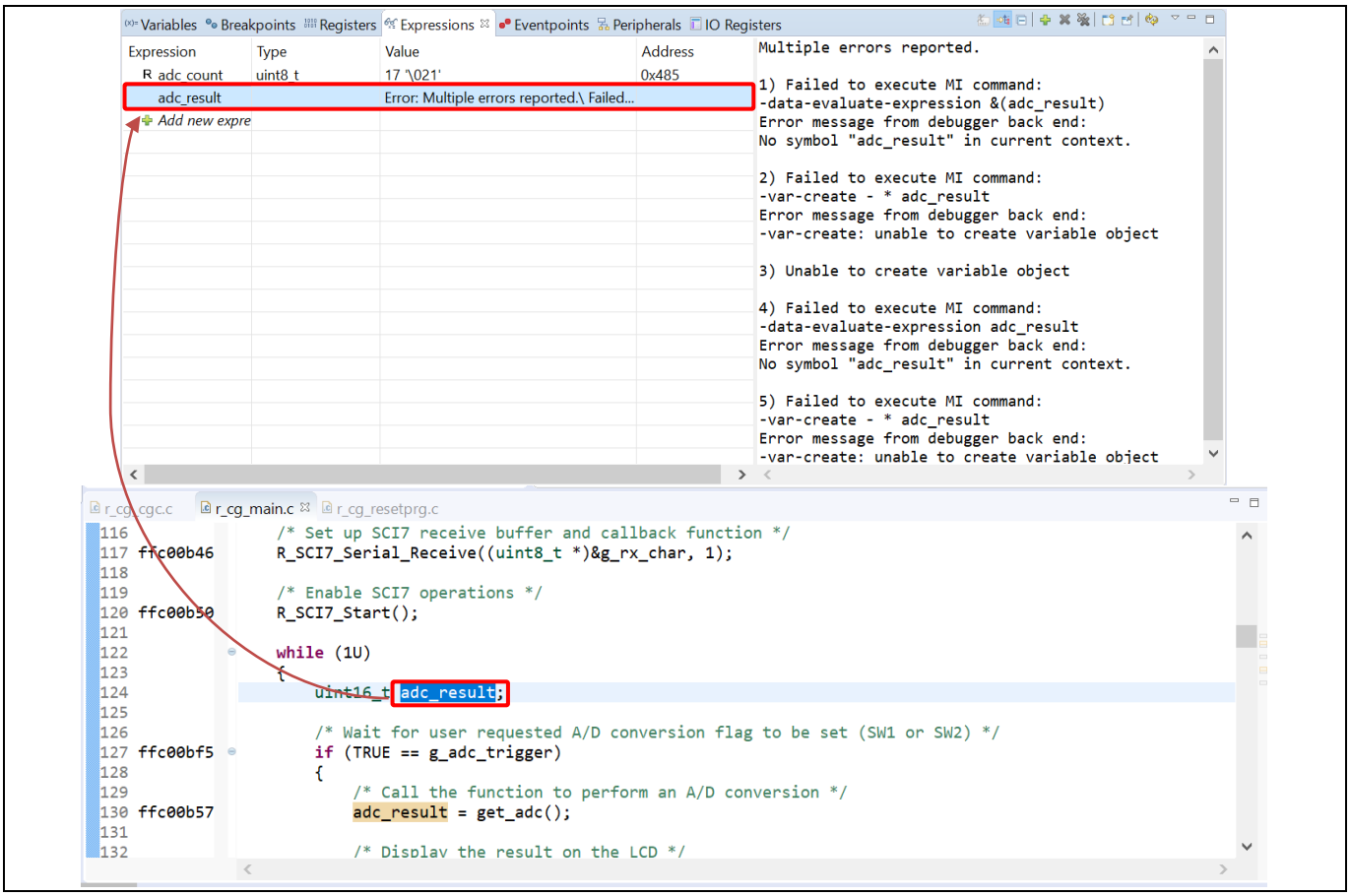


Figure 5-14 Add local variable to Expression view

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.

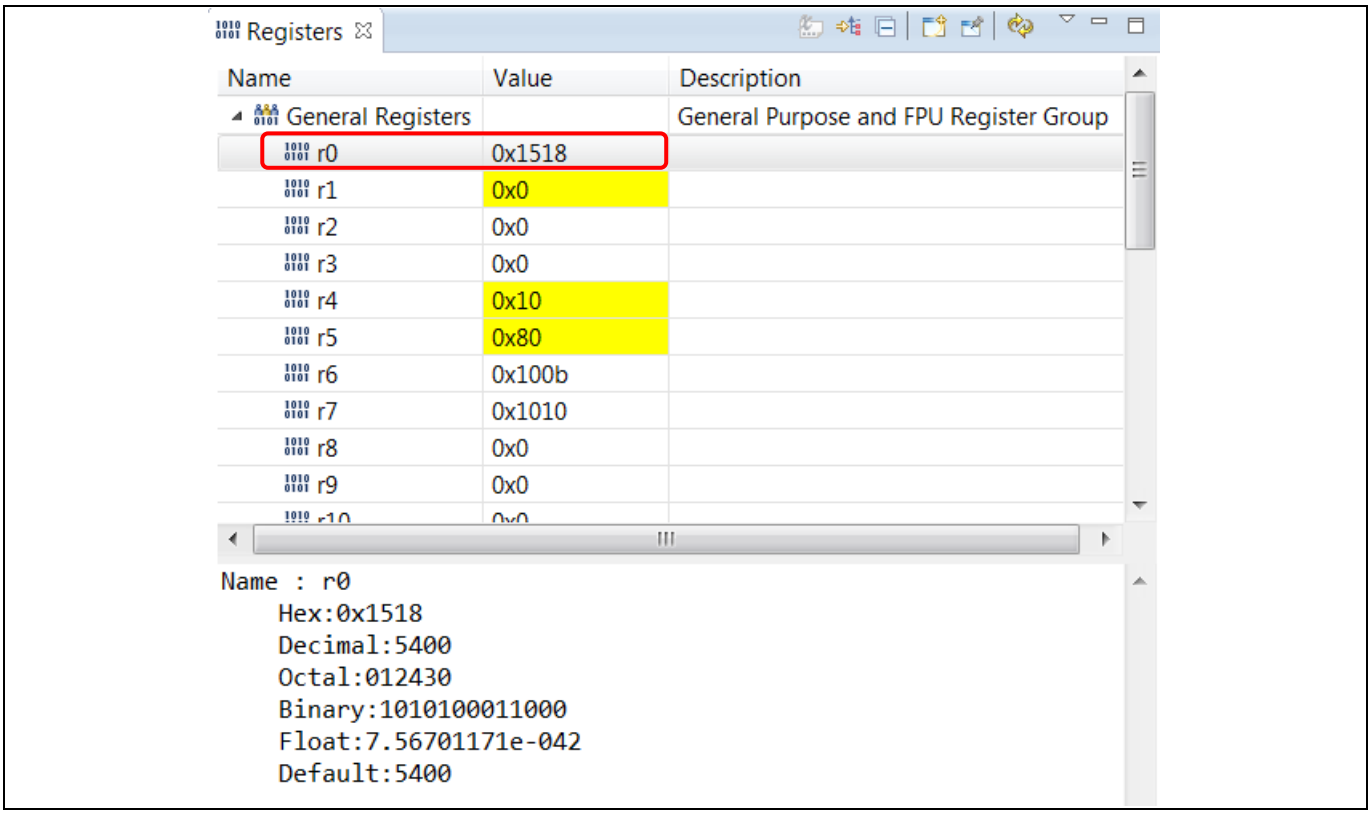


Figure 5-15 [Registers] View

To view the general register “r0”,

1. Click [Window] → [Show View] → [Registers] or icon 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”.

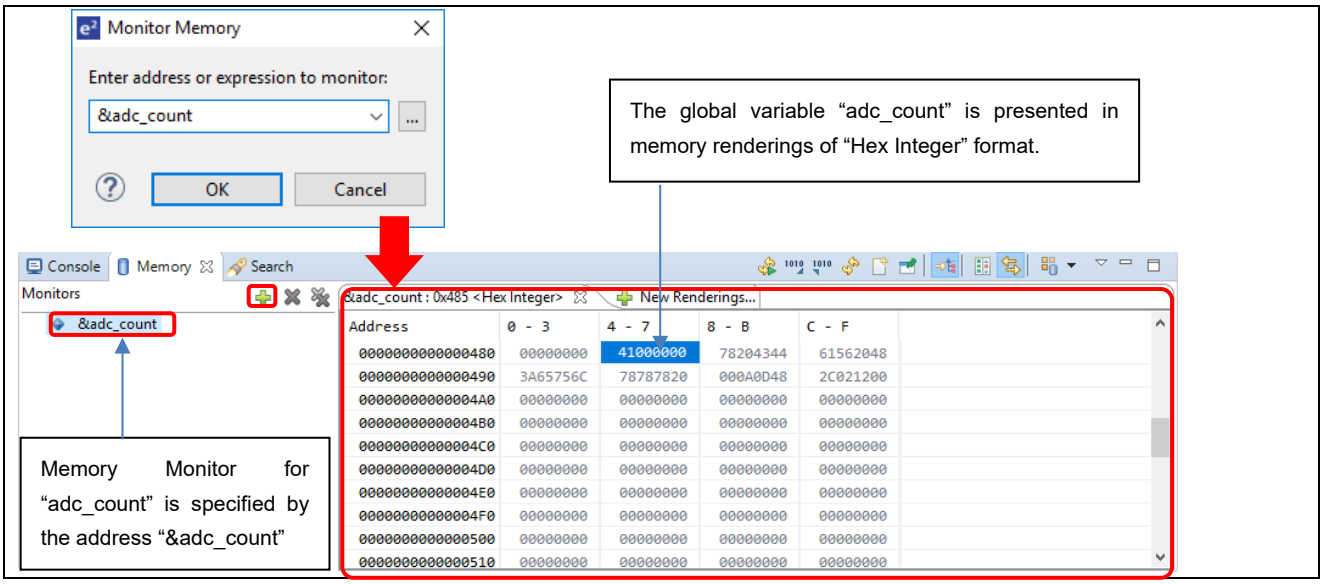
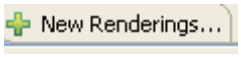


Figure 5-16 [Memory] View (1/2)

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

1. Click the tab 

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>”.

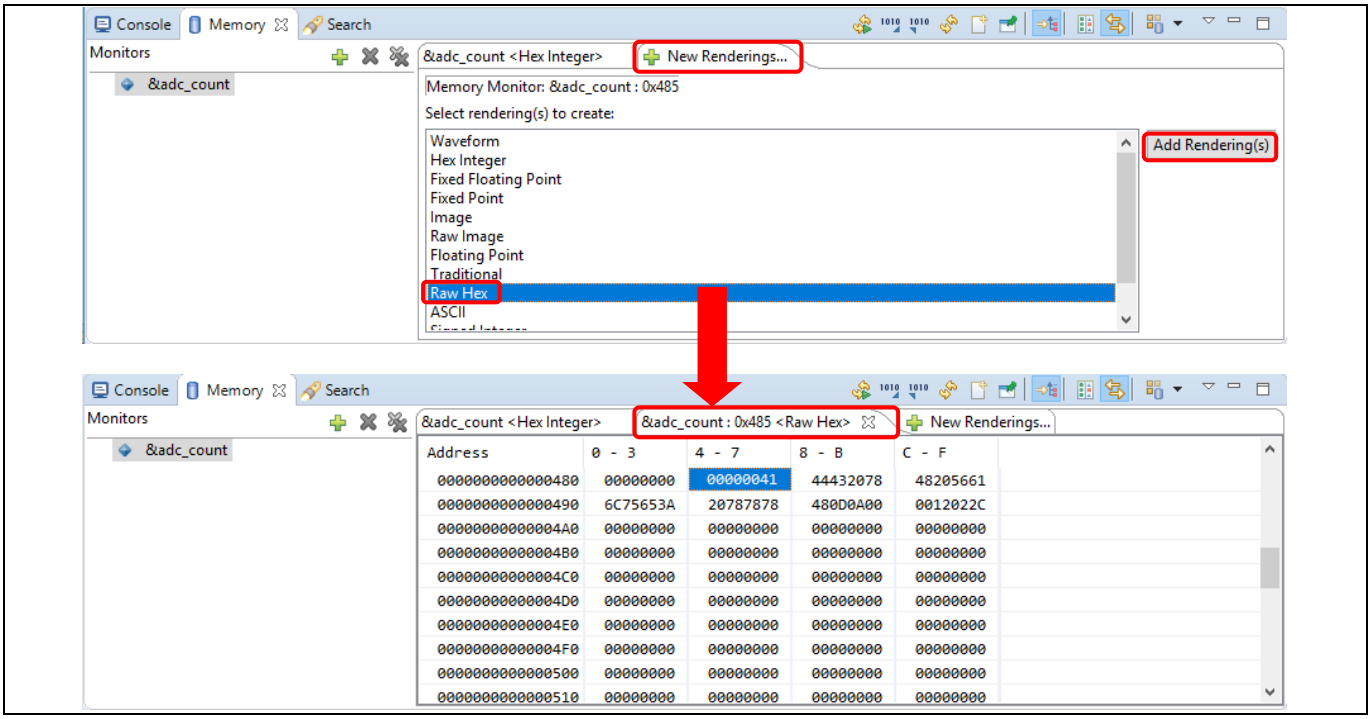


Figure 5-17 [Memory] View (2/2)

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.

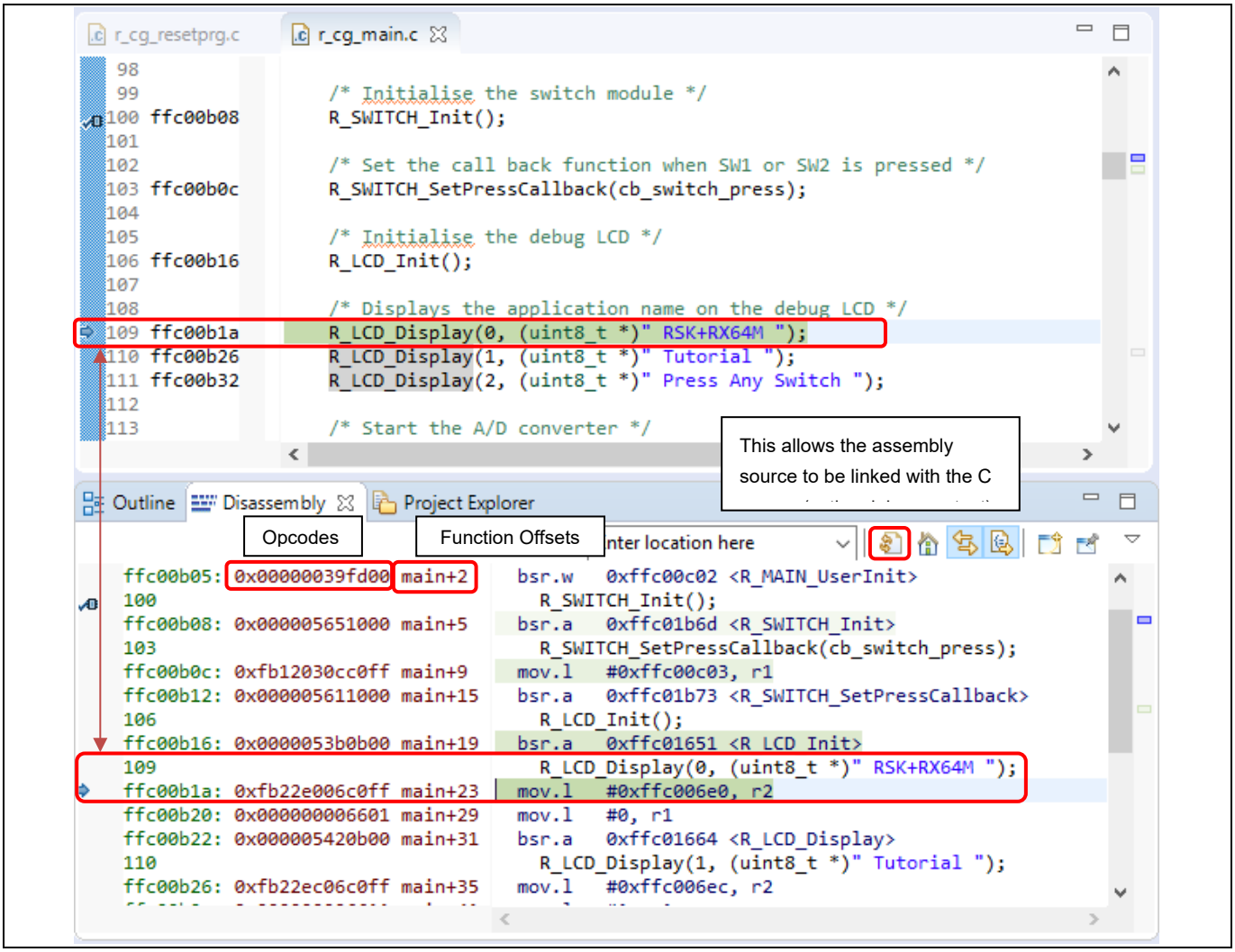


Figure 5-18 [Disassembly] View

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.

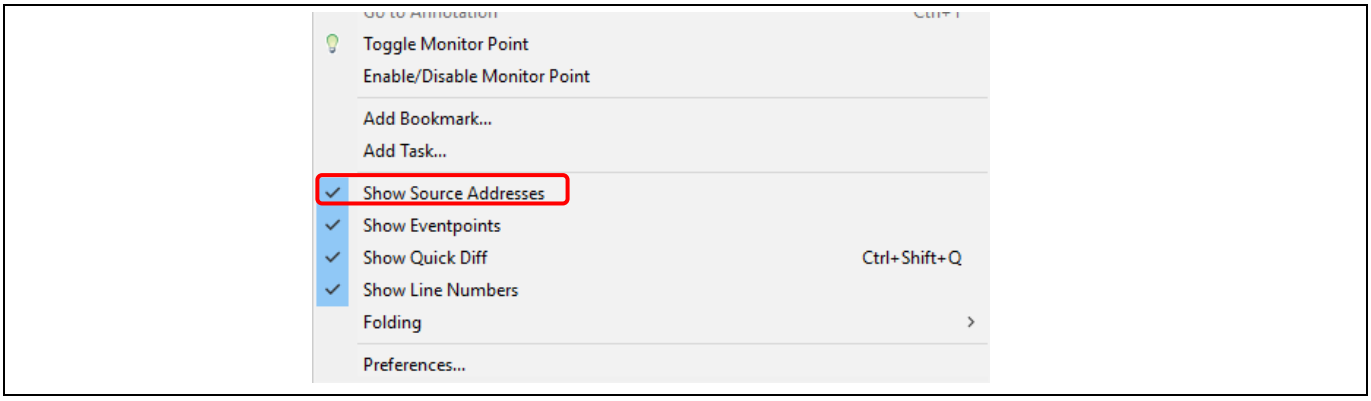


Figure 5-19 Source Addresses Menu

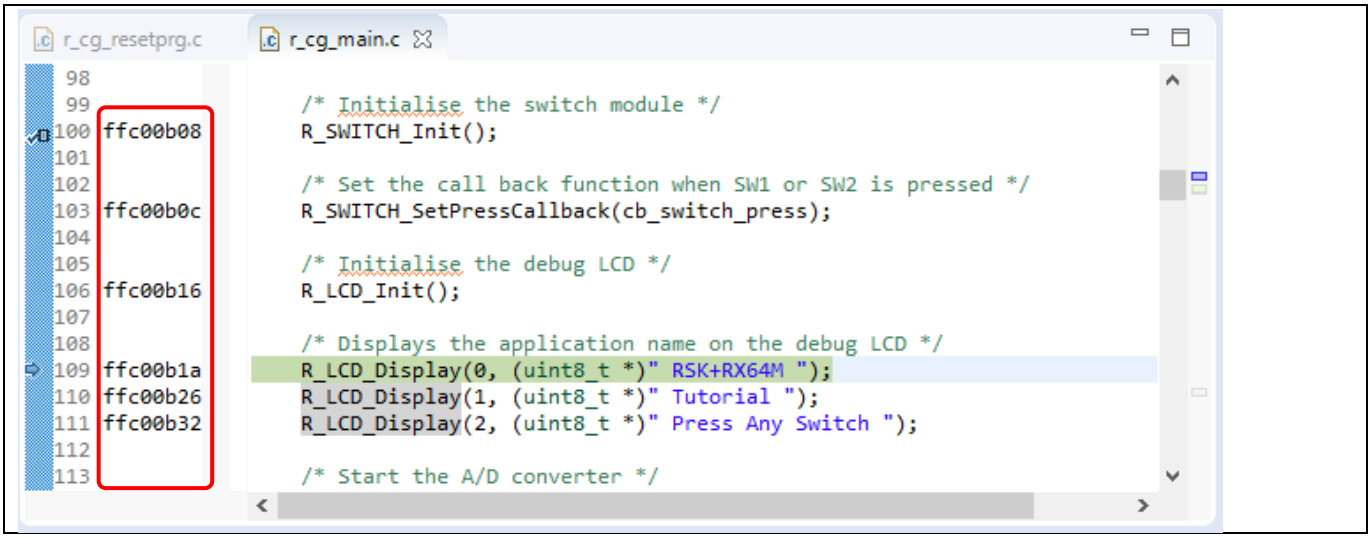


Figure 5-20 Source Addresses displayed in Editor

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.

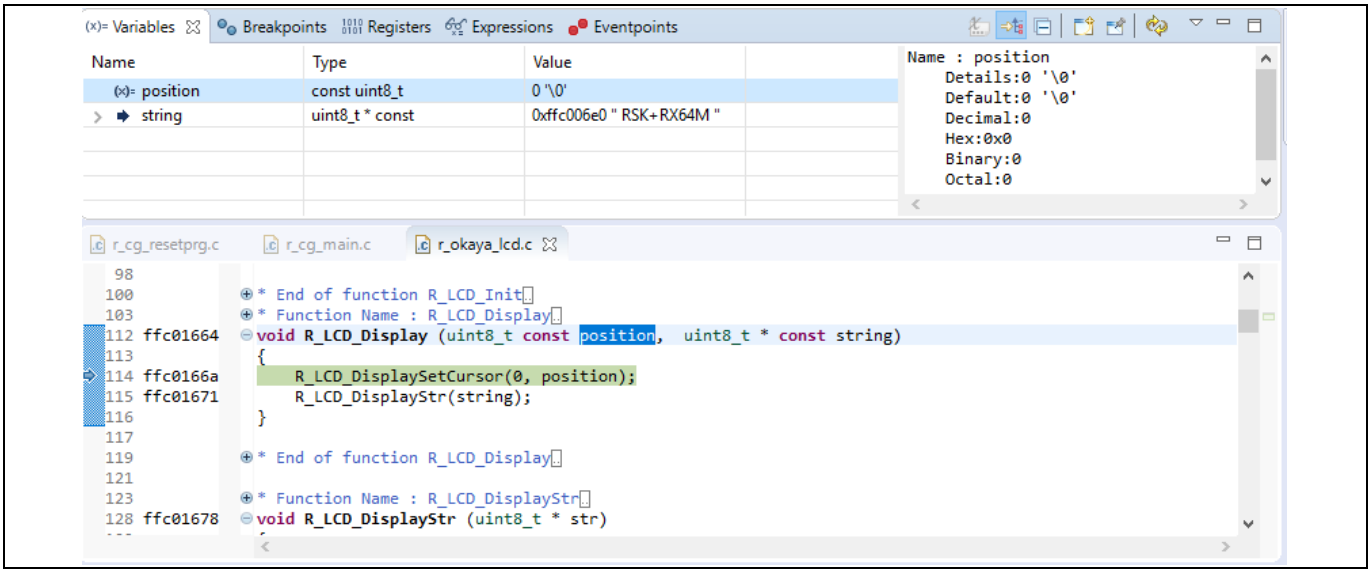


Figure 5-21 [Variables] View

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.
By disabling optimization, variables would become visible in most of the cases. However, it means to give up all benefits of optimization such as memory efficiency, code size reduction and performance improvement.

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.

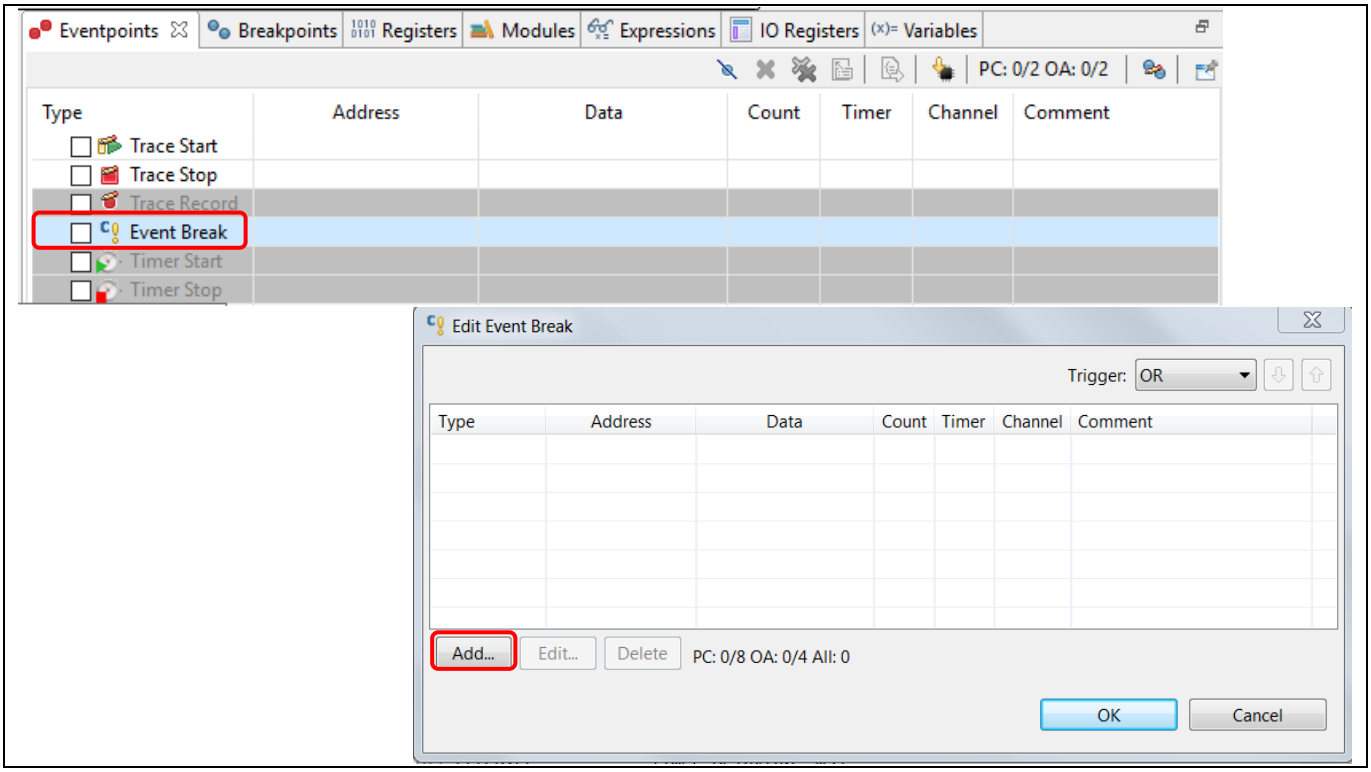


Figure 5-22 [Eventpoints] View (1/2)

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.

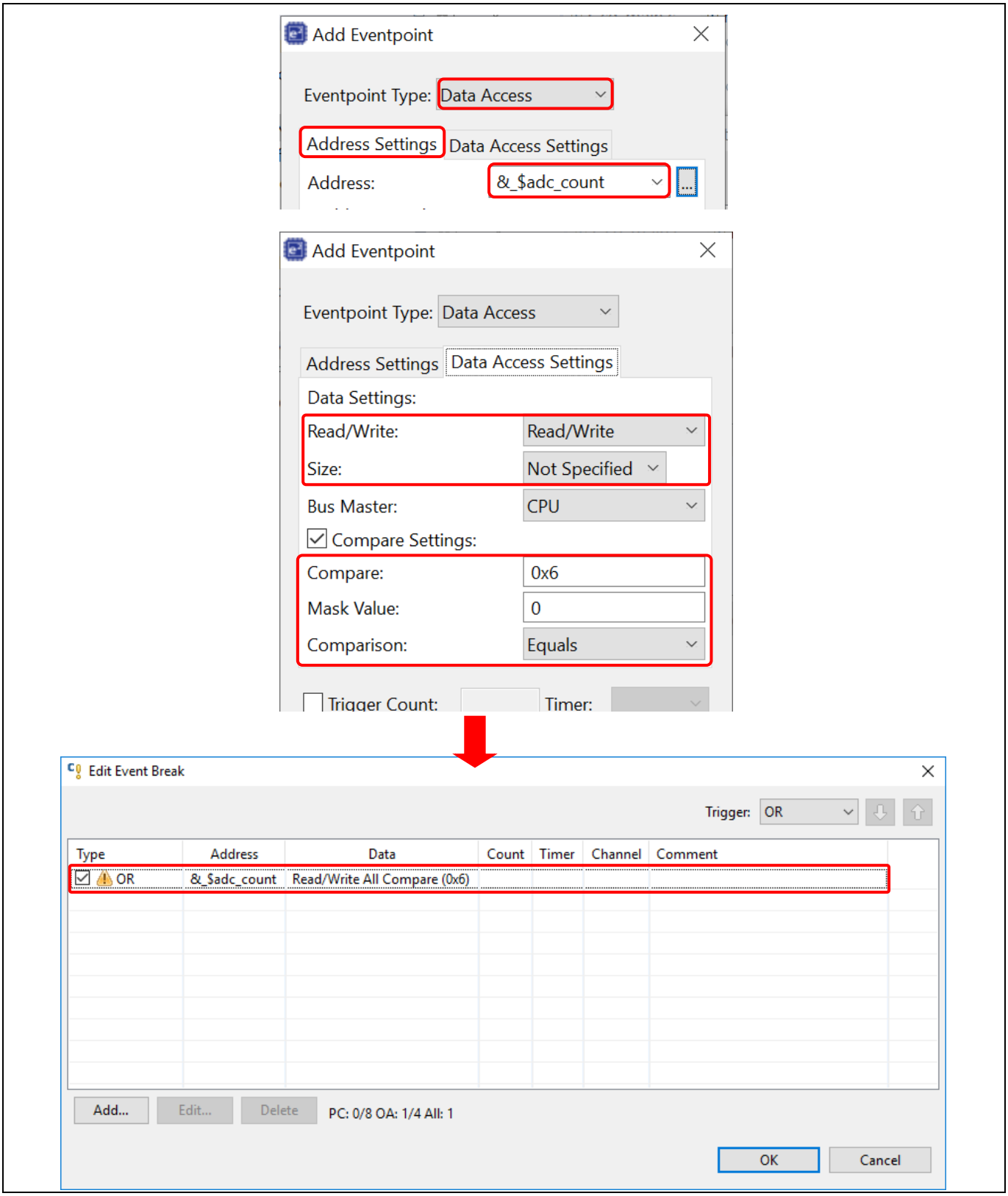


Figure 5-23 [Eventpoints] View (2/2)

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.

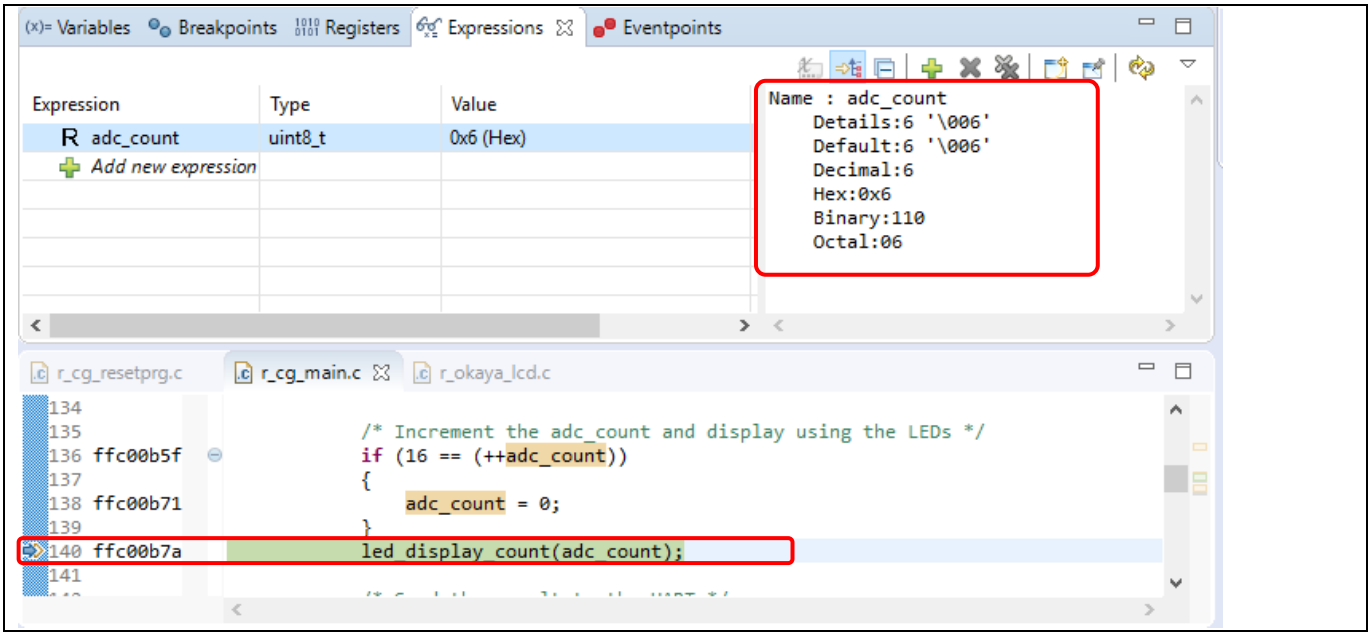


Figure 5-24 Execution of Event Break

Figure 5-24 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.

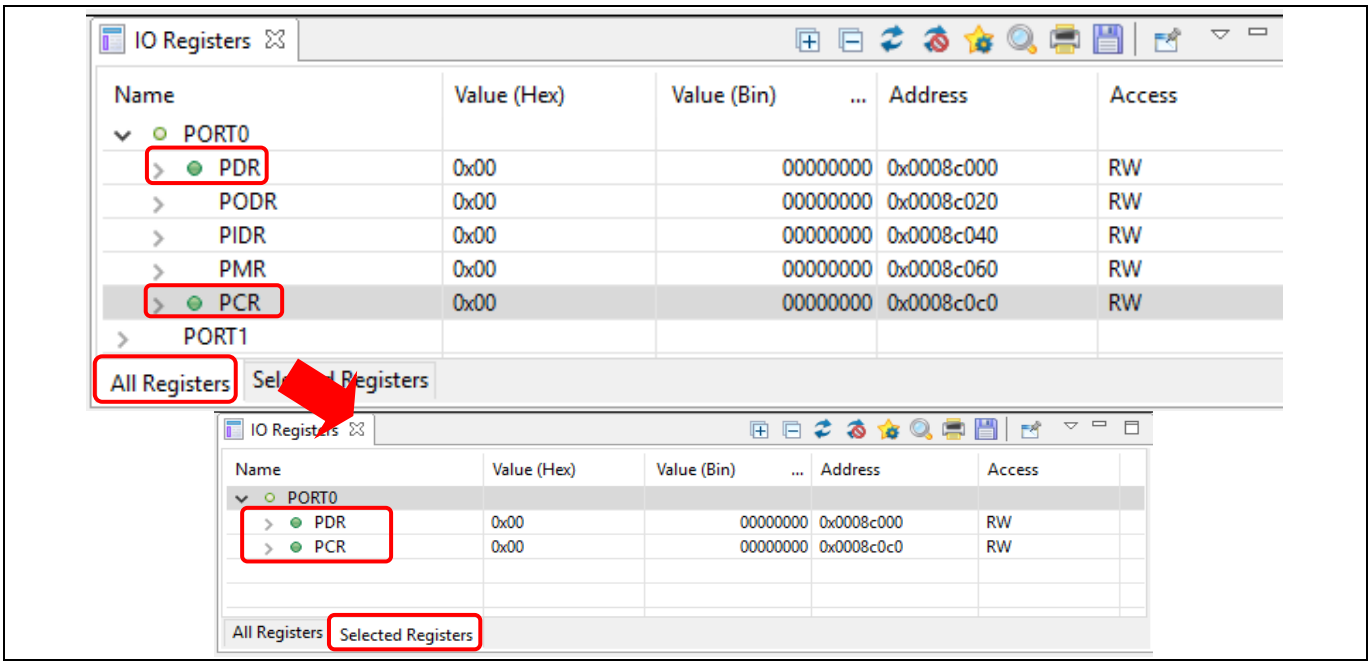


Figure 5-25 [IO Registers] View

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.
   You could also use Search button in the IO Register toolbar to quickly search by name.
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.

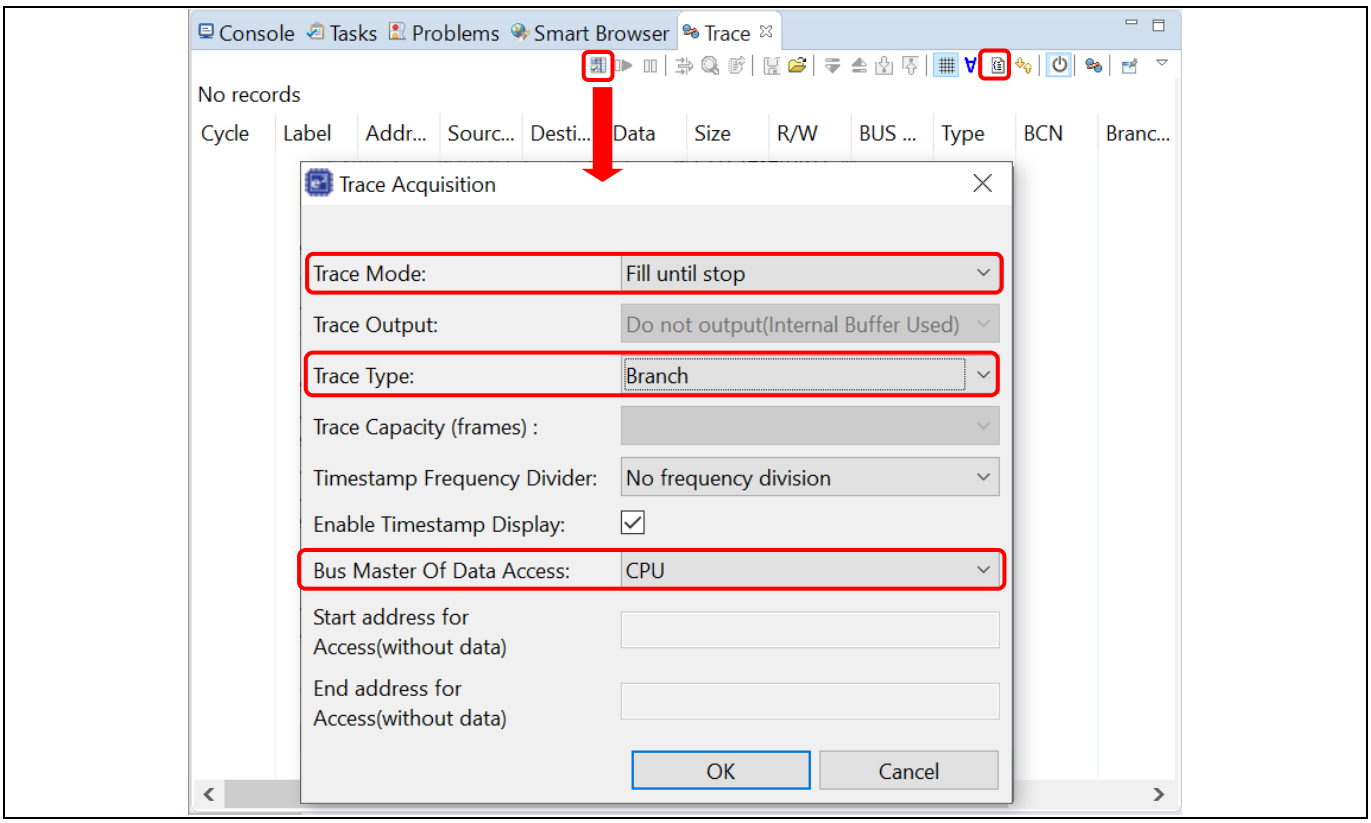


Figure 5-26 [Trace] View (1/2)

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

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

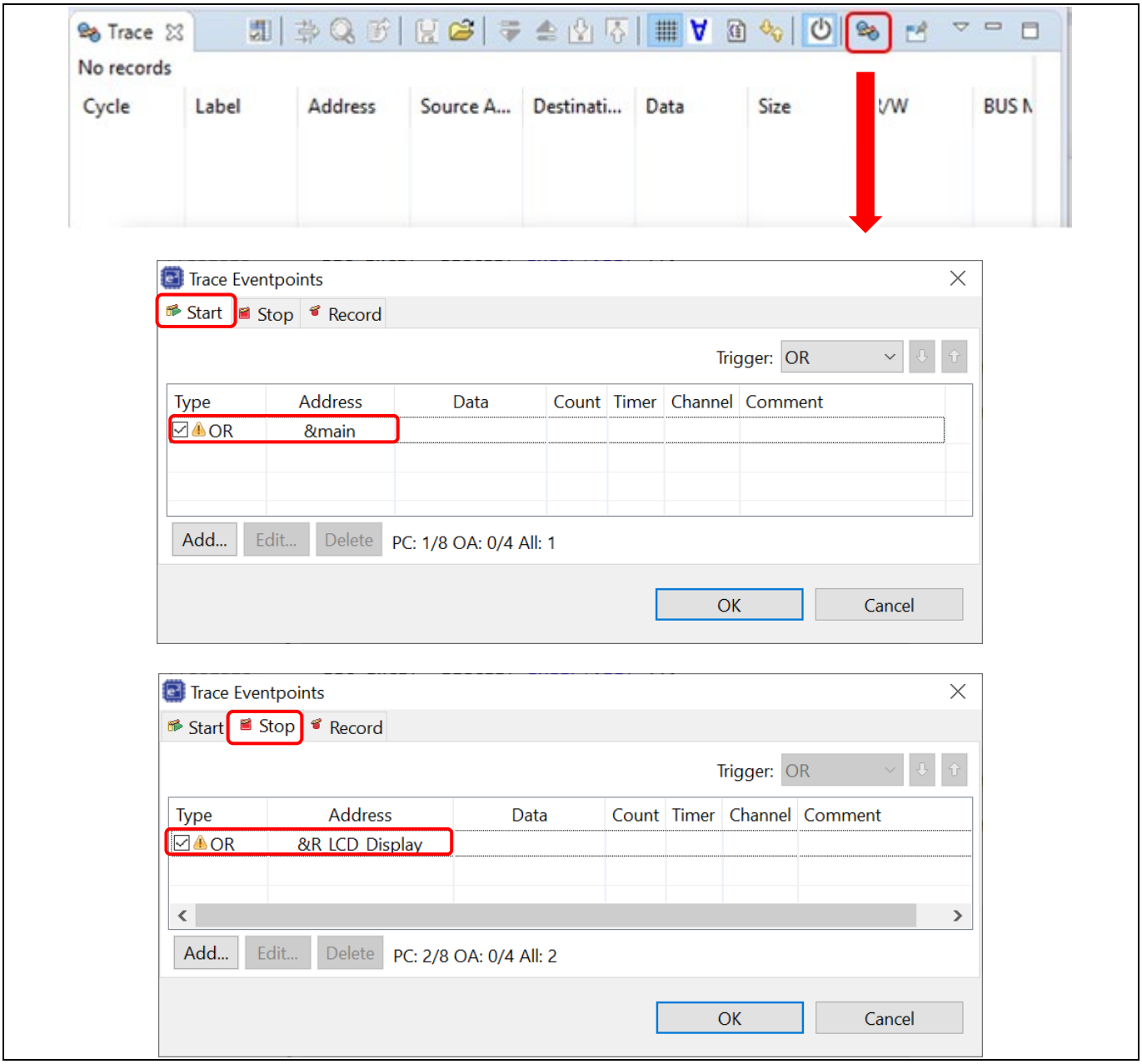


Figure 5-27 [Trace] View (2/2)

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

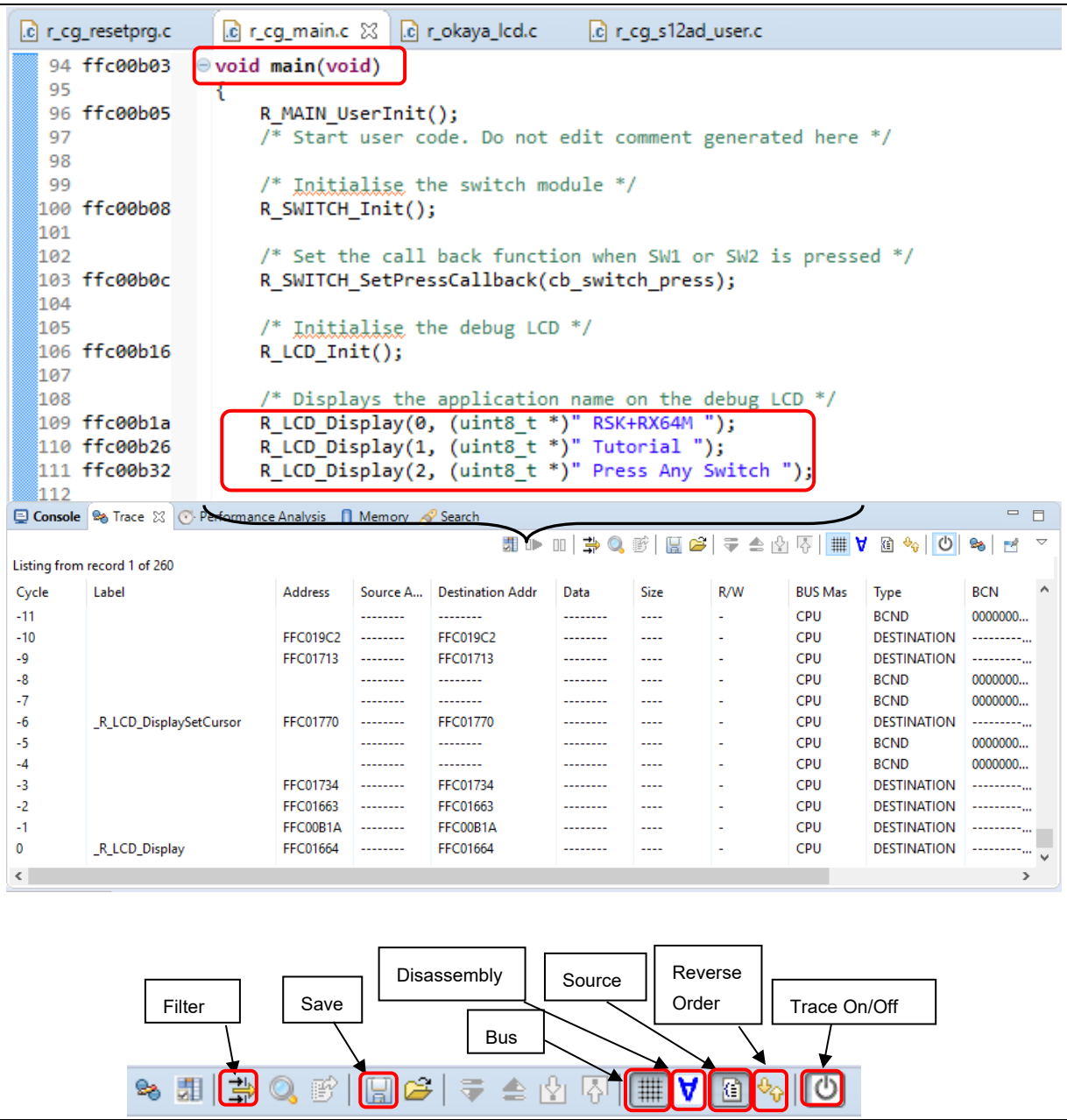


Figure 5-28 Point-to-Point Trace between Two Functions

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/search/keyword-search.html#q=R20UT0399

5.4.10 Memory Usage View

Memory view allows users to view the total memory size, usage of ROM and RAM ratio and detailed information of sections, objects, symbols, module, vector and cross reference used in project.

To view the memory usage of a project,

1. Click [Window] → [Show View] → [Other…] → [Debug] → [Memory Usage] to open the memory usage view.
2. The default display of memory usage view is different for each kind of project.
   a. The GUI of memory usage view for executable project which uses Renesas Toolchain includes 3 regions: (1) Group size region, (2) RAM/ROM Usage region/Device Memory Usage region, (3) Detail table region. The map file location is shown at bottom bar.

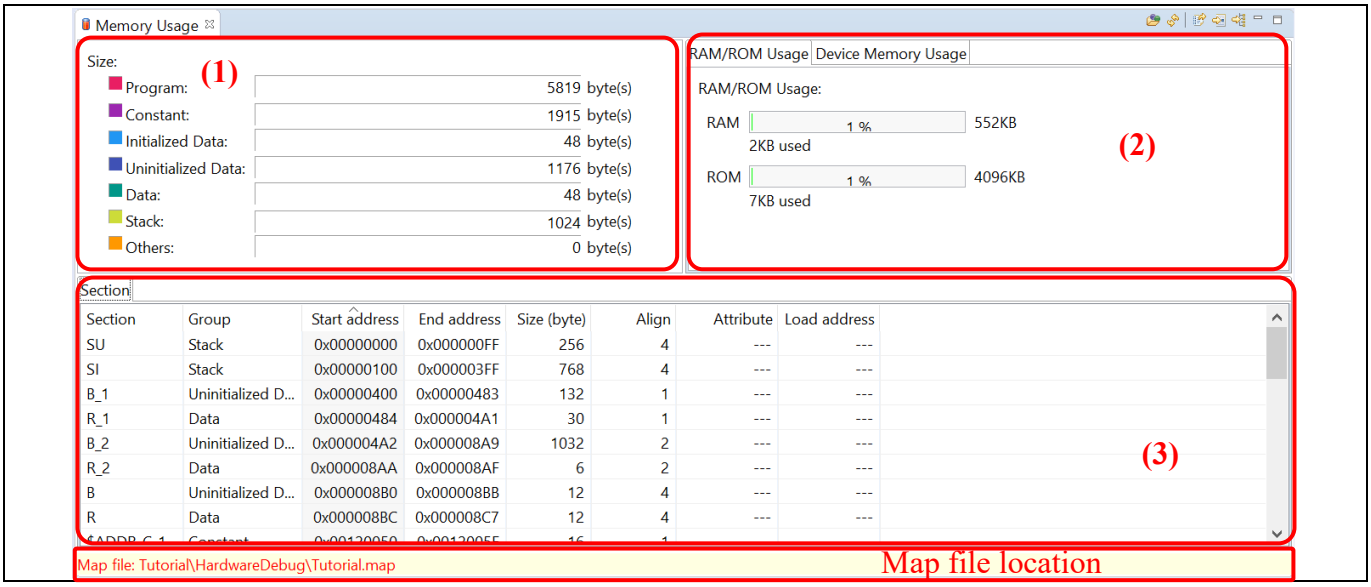


Figure 5-29 Executable Project - Renesas Toolchain

   b. For executable project which uses GCC Toolchain, “Memory region usage" region (2) will be displayed instead of RAM/ROM usage region:

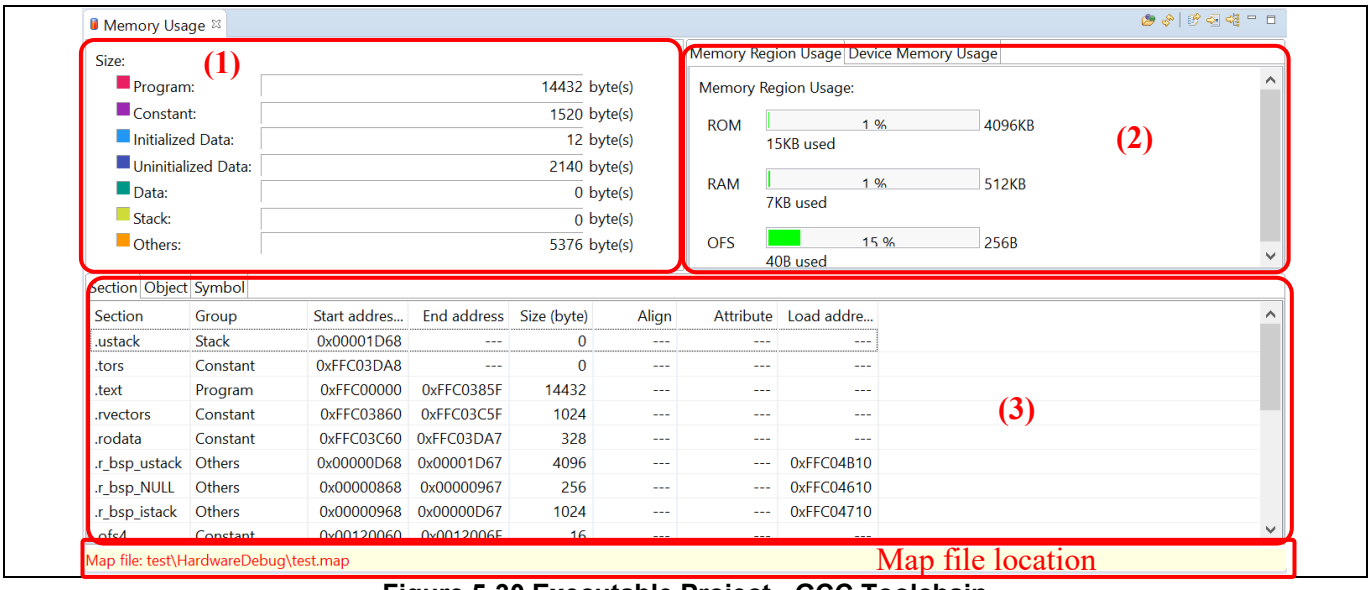


Figure 5-30 Executable Project - GCC Toolchain

c. For library project which uses Renesas Toolchain, “Library information view" will be displayed instead of “Group size view”

Note: Only available for Renesas CCRX or CCRL toolchains.

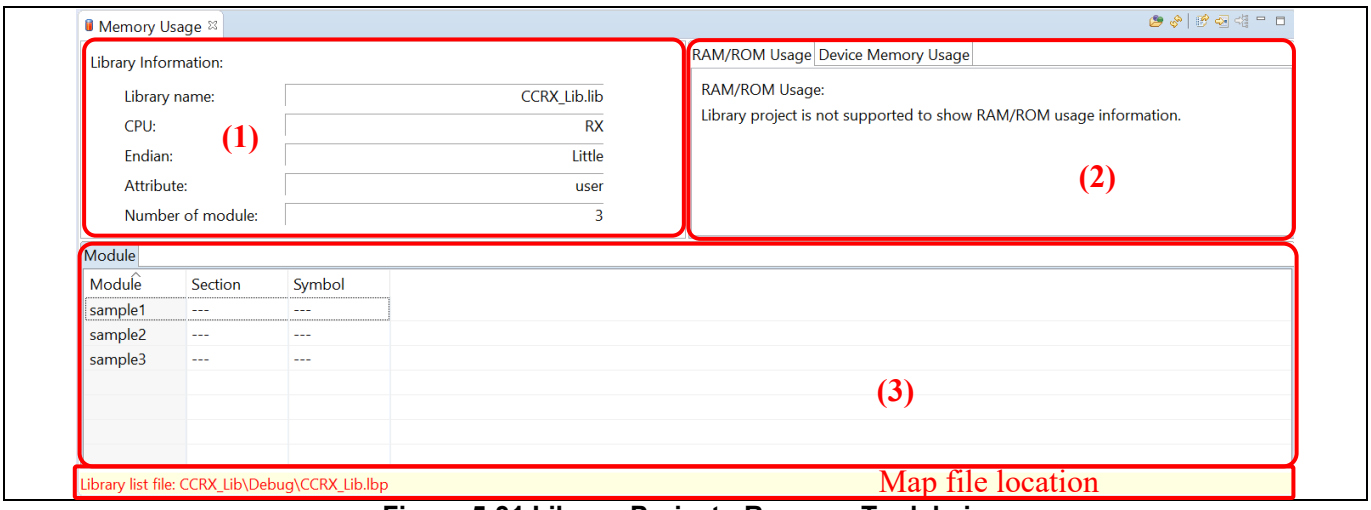


Figure 5-31 Library Project - Renesas Toolchain

d. Memory usage view is not available for library project which uses GCC Toolchain.

Group size view:

Displays the total size of Program, Constant, Initialized Data, Uninitialized Data, Data, Stack and Other according to the selected map file.

Note: This view only displays for executable project of supported toolchains.

Library information view:

Displays the information of selected library list file. The information to be visualized on this region consists of:

- The name of the selected library
- The type of CPU specified by the project
- Endian
- Attribute
- Number of module.

RAM/ROM usage region:

Shows percentage of RAM/ROM usage by numerical value and status of bar. Color of bar is based on percentage value.

- If percentage < 75%: Green.
- If percentage >= 75% and percentage < 90%: Orange.
- If percentage >= 90%: Red.

Memory region usage region:

Display feature of this region is similar to RAM/ROM usage region.

Device memory usage region:

Device Memory Usage region shows device memory of selected project's device. Each memory area shows name, start address, end address, used size and size.

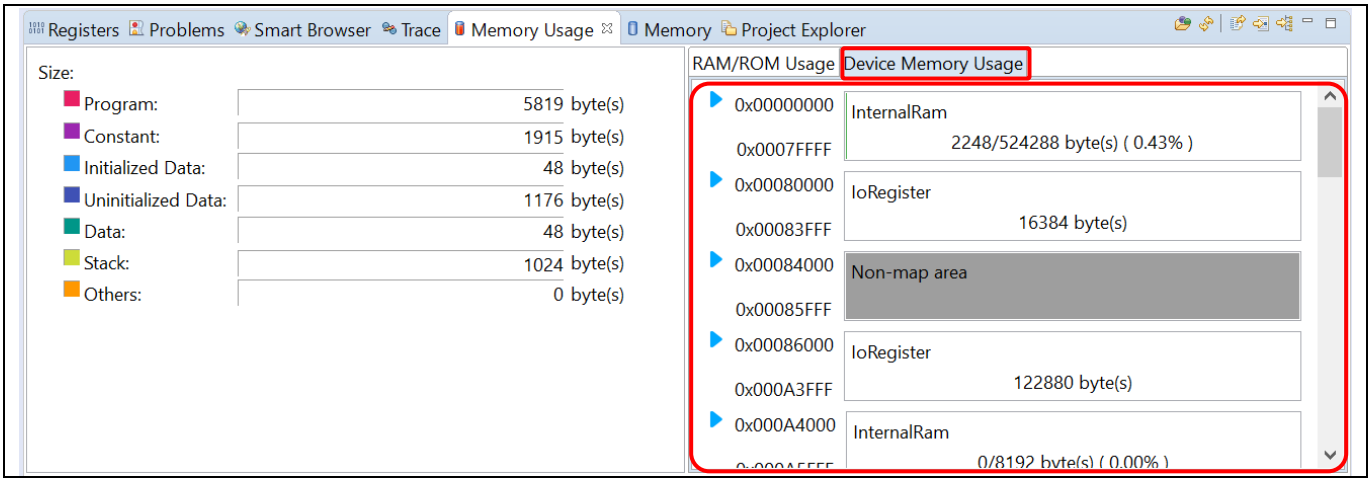


Figure 5-32 Device memory usage region

Expand memory area to see all sections. Color of sections corresponds with that of Group Size Region.

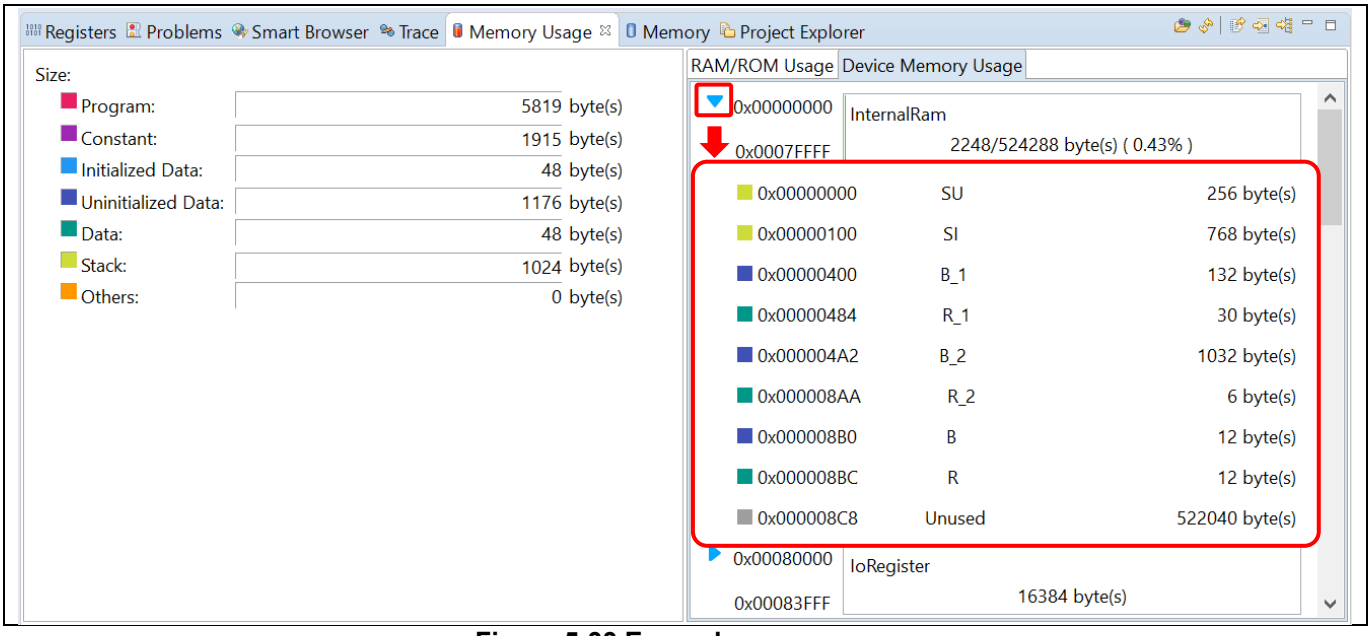


Figure 5-33 Expand memory area

Detail table region:

Display the map file information of an active project or the opened map file.

- "Section" tab: contains "Linkage map" table which displays the list of Sections analyzed from map file and its detailed information.
- "Object" tab: contains "Object" table which displays the list of Objects analyzed from map file and its detailed information.
- "Symbol" tab: contains "Symbol" table which displays the list of Symbols analyzed from map file and its detailed information.
- "Vector" tab: displays the vector table information that is retrieved from map file. This tab is only available for executable project that is configured to work with these toolchains: Renesas CCRX, Renesas CCRL.

- “Cross Reference” tab: displays the cross reference information that is retrieved from map file. This tab is only available for executable project.
- “Module” tab: contains “Module” table. “Module” tab is only available for library project that is configured to work with these toolchains: Renesas CCRX, Renesas CCRL.

Map file location:

Memory Usage will display the information of (*.map) file or library list file (*.lbp) from project. User can see the relative path of selected map file or library list file at the bottom of Memory Usage view.

6. Help

The help system allows users to browse, search, bookmark and print help documentation from a separate Help window or Help view within the workbench. Users can also access an online forum dedicated to the e² studio from here.

Click on [Help] tab to open the Help menu.

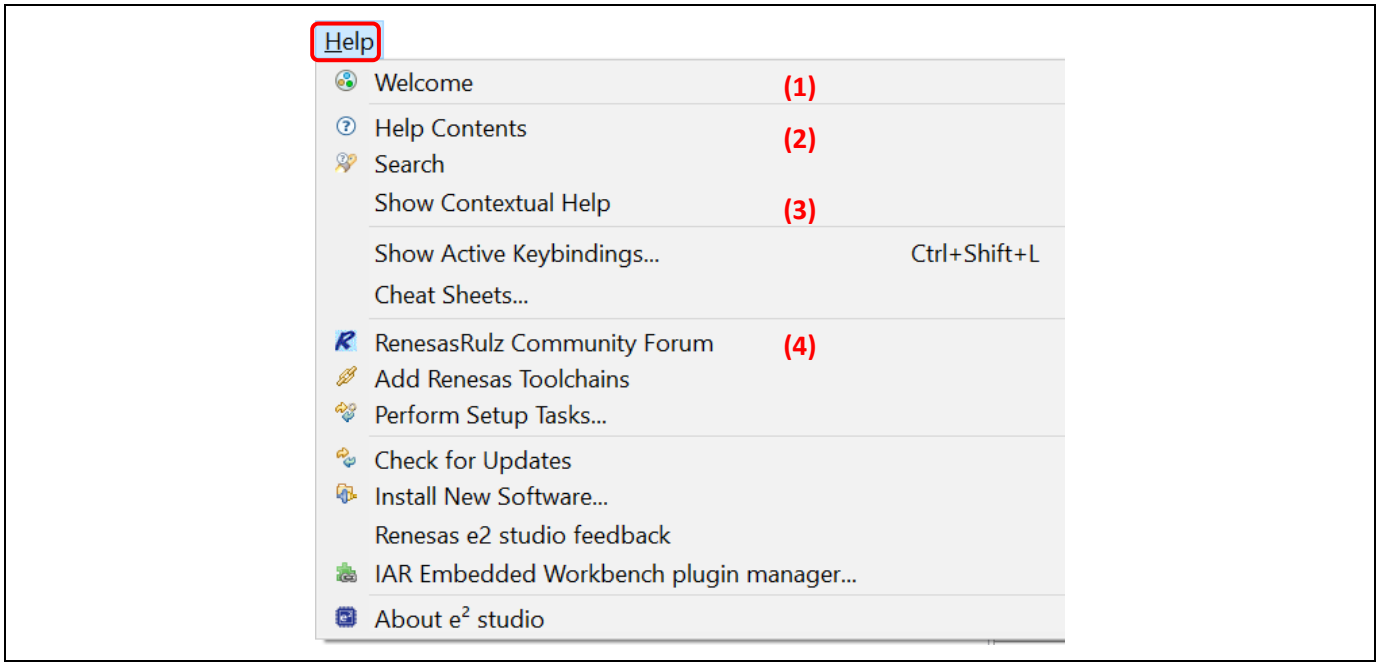


Figure 6-1 Help – Help Menu

Quick Help Tips:

(1) Click [Welcome] for an overview of the e² studio and to view Release Notes.

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

There are many useful topics under the [Help Contents]. Example, the “Debugging Projects” topic which provides useful information such as debug configuration, supported number of breakpoints, etc. It can be launched by clicking on the [Help] menu → [Help Contents] → “e² studio User Guide”.

(3) Click [Show Contextual Help] to open the Help view within the workbench.

(4) Click [RenesasRulz Community Forum] to go an online forum that is dedicated to topics and discussions related to the e² studio (Internet connection is required).

Revision History	Renesas e^2 studio User's Manual: Getting Started Guide

Rev.	Date	Description	
		Page	Summary
1.00	May 30, 2020	-	First edition for RX, RL78, RH850 based on e^2 studio 2020-04 and v7.8.0

Renesas e^2 studio User's Manual: Getting Started Guide for RX, RL78, RH850 Family

Publication Date: Rev.1.00 May.30.2020

Published by: Renesas Electronics Corporation

RX/RL/RH Family

Renesas Electronics Corporation

R20UT4819EJ0100