User's Manual

e² studio

Integrated Development Environment

User's Manual: Quick Start Guide

RENESAS MCU
RX Family, RL78 Family, RH850 Family,
RISC-V MCU and DA Devices

Renesas Electronics
www.renesas.com

Rev.1.20 Feb 2025

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 be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. 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.
6. 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.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. 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.5.0-1 October 2020)

Corporate Headquarters

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

www.renesas.com

Trademarks

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

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/.

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.

How to Use This Manual

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

e^2 studio, which is an integrated development environment (IDE) for the Renesas device families (RX, RL78, RH850, RA, RE, RZ, RISC-V MCU, and DA), has the necessary tools for the phases (design, implementation, and debugging) of software development integrated into a single platform.

Providing an integrated environment allows all stages of development to be performed with the use of this product alone, eliminating the need to use many different tools.

Readers

This manual is intended for users who wish to use e^2 studio in developing software or hardware application systems and for users of the RX, RL78, RH850, RISC-V MCU, and DA.
For users of other device families, refer to the user's manuals for the individual device families.

Purpose

This manual aims to provide users with explanations of the functions provided by e^2 studio when they commence the development of hardware or software systems for use on target devices.
For software packages (FIT, FSP, etc.) and code generators (Smart Configurator and FSP Configurator), refer to the individual user's guides.

Structure

This manual can be broadly divided into the following sections.

Section 1	General
Section 2	Installation
Section 3	Project Generation
Section 4	Build
Section 5	Debug
Section 6	Help

How to read this manual

It is assumed that the readers of this manual have general knowledge of PCs 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
Remarks:	Supplementary information
Numeric representation:	Decimal ... XXXX Hexadecimal ... 0xXXXX

Table of Contents

1. General

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 document describes the usage of e² studio IDE to develop applications for the RX family series microcontrollers as an example.

1.1 System Configuration

Below is an example of a typical system configuration.

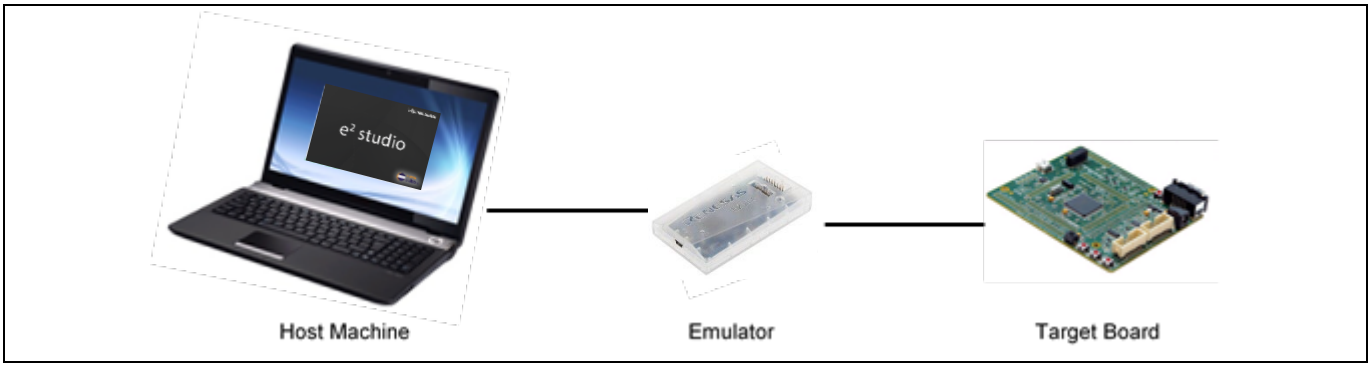


Figure 1-1 System Configuration

1.2 System Requirements

Hardware Environment:

Processor:	x64-based processor, at least 2GHz (support hyper-threading/multi-core CPU)
Main Memory:	At least 4GB, recommended 8GB of free memory space.
Hard disk 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.

[Windows PC]

Supported OS: Windows 10 and Windows 11 (64-bit edition)
(32-bit Windows is supported with e² studio V7.8.0)

Microsoft Visual C++ 2010 SP1 runtime library*
Microsoft Visual C++ 2015-2022 runtime library*
*: Installation of the library is at the same time as that of e² studio.

[Linux PC]

Supported OS: Ubuntu 20.04 LTS Desktop (64-bit version) and later versions

[macOS PC] (Apple Silicon)
macOS 13 (Ventura) and later versions

Refer to the release note for details on the supported OS versions and features of the Linux and macOS versions. Operation may not be as intended with an OS version that is not supported. Linux and macOS versions of compiler products such as CC-RX, CC-RL, and CC-RH may not be supported or may have non-supported features. In such cases, use the given compiler product in a Windows environment. For details, refer to the release note for the given compiler product.

This document is described based on the Windows version. Please refer to e² studio Release Note or FAQ page (mentioned in the Release Note) for installation in the Linux PC. Operations after installation are very similar to those on the Windows version.

1.3 Supported Toolchains

Renesas C/C++ compiler package for RX family (CC-RX)
Renesas C compiler package for RL78 family (CC-RL)
Renesas C compiler package for RH850 family (CC-RH)
GCC for Renesas RX Toolchain
LLVM and GCC for Renesas RL78 Toolchain
LLVM for Renesas RISC-V MCU Toolchain

Note 1: Two types of packages (with CS+ and without IDE) for each CC-RX, CC-RL and CC-RH 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.

Note 3: For DA devices, supported toolchains differ according to the SDK project. Building with GCC for Arm (including MinGW GCC) and LLVM for Arm is possible with e² studio. Check the user's manual for the SDK, register the appropriate toolchain with e² studio, and import the project. Refer to section 2.4, Installation of Compiler Packages.

1.4 Supported Emulator Devices

E2 emulator Lite (RX, RL78, RISC-V MCU)

E1 (RX, RL78, RH850) (Note 1)

E2 (RX, RL78, RH850, RISC-V MCU)

E20 (RX) (Note 2)

Segger J-Link (RX, DA, RISC-V MCU)

Note 1: The E1 emulator is EOL (end-of-life) product.
Note 2: The E20 emulator does not support RH850-family MCUs with the next-generation G4MH core, such as those of the RH850/E2x series. The E2 emulator is available for use with them.

1.5 Supported Simulators

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

2. Installation

The latest e² studio IDE installer package can be downloaded from Renesas website for free. It is recommended to install the 64-bit version on 64-bit PC. The e² studio installers and related documentation are available on the e² studio product page https://www.renesas.com/e2studio, on the device-family information page. Note that users must log in to the Renesas account (on MyRenesas page) for the software download.

e² studio is available in 32-bit (V7.8.0 or earlier) and 64-bit (2020-04 or later). Use the 64-bit e² studio versions unless in the 32-bit OS environment.

This chapter describes the installation and uninstallation for the e² studio IDE.

Please uninstall the earlier versions before installation. Alternatively, install new e² studio into a new folder if you would like to keep earlier versions. Although upgrade installation is available, upgrading between incompatible versions is not allowed such as from V7.8.0 to 2020-04, 2024-01 to 2024-04. Please install it into a new folder if fails.

The detailed information is described below.

2.1 Installation of the e² studio IDE (64-bit Version)

1. Launch the e² studio installer. For e² studio 2023-07 and later, there are three (Lite/Standard/Custom) types of new installations. Select any of the items and click [Install].

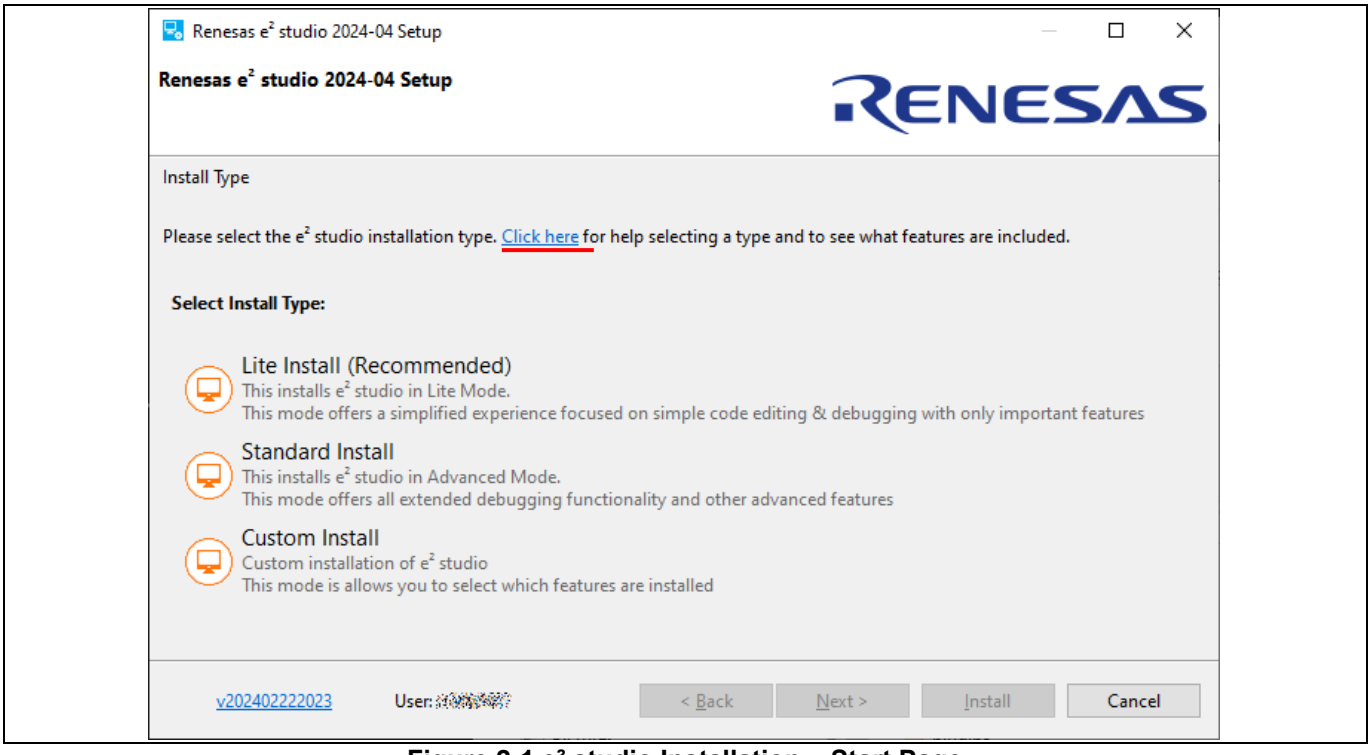


Figure 2-1 e² studio Installation – Start Page

Lite installs the minimum functionality required to build and debug your project. In addition, Standard can install extended functionality. Any function can be installed in Custom. Clicking on the "**Click here**" link displayed on the first page of the installer will show a list of the functions that will be installed by Lite and Standard, so please confirm it before proceeding.

Note: If the e² studio is already installed, the [Modify] (modifying installed e² studio) and [Uninstall] options are displayed in addition to [Install].

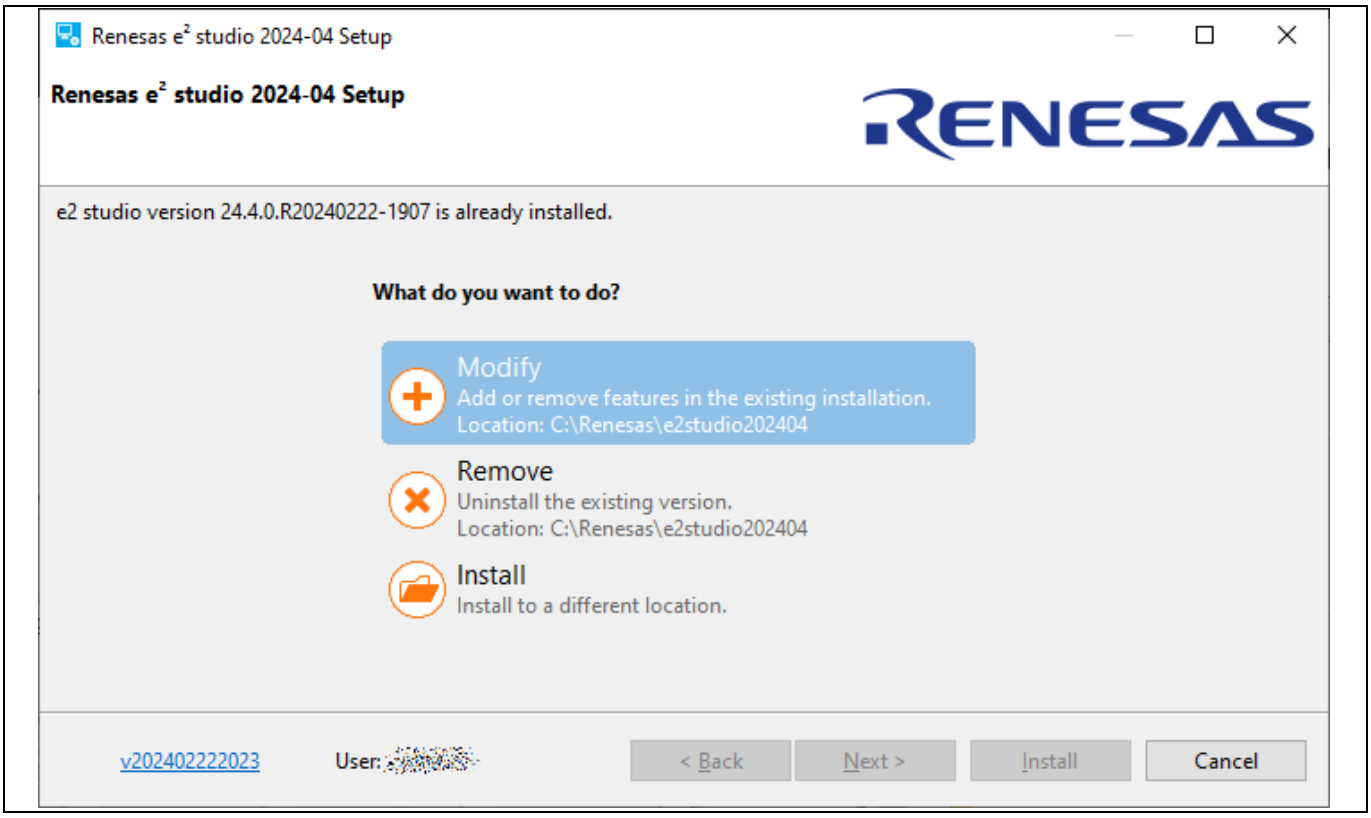


Figure 2-2 e² studio Installation – Detected Installation of e² studio

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 the e² studio installation folder name.

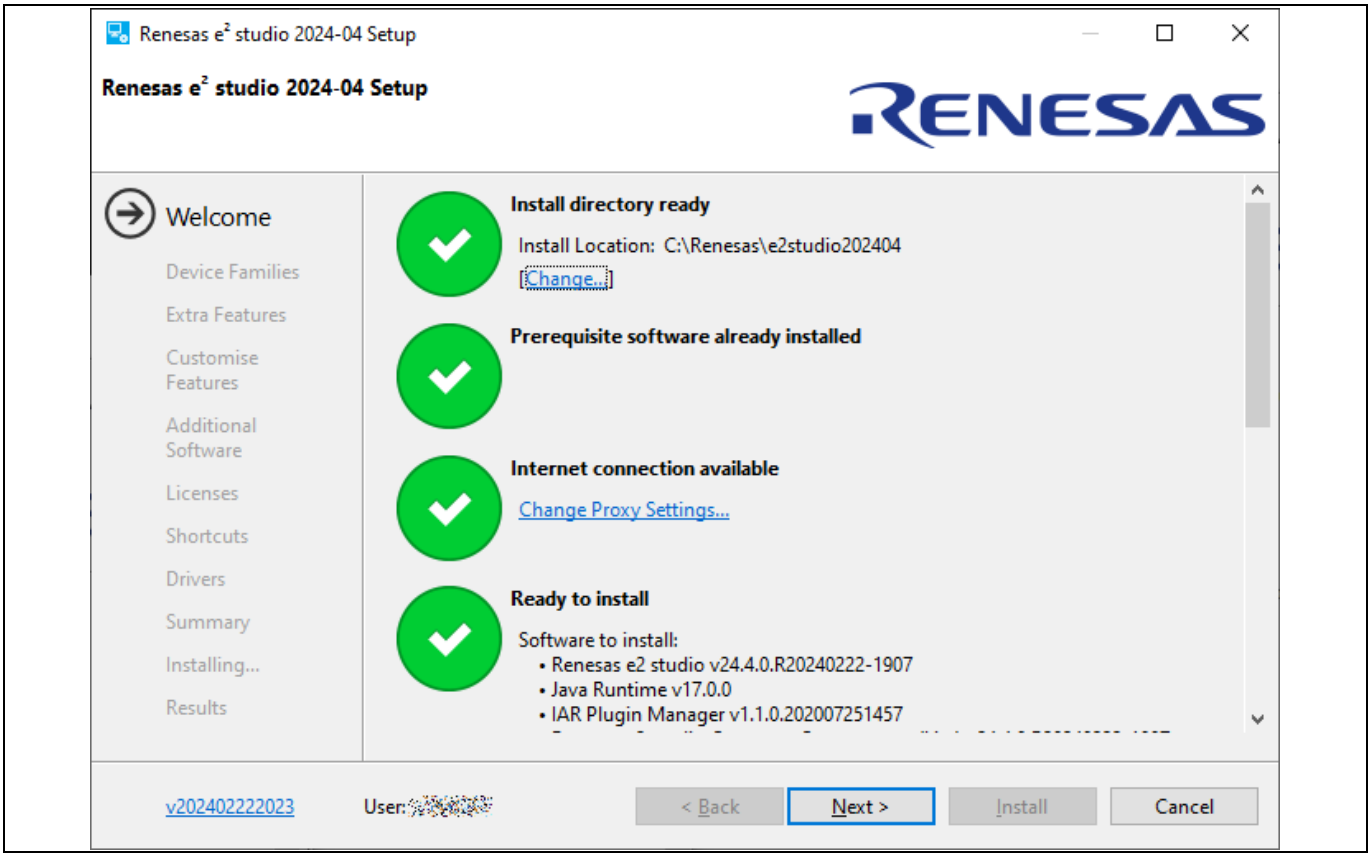


Figure 2-3 e² studio Installation – Welcome Page

If the installer cannot access Renesas web, a warning message will appear as follows:

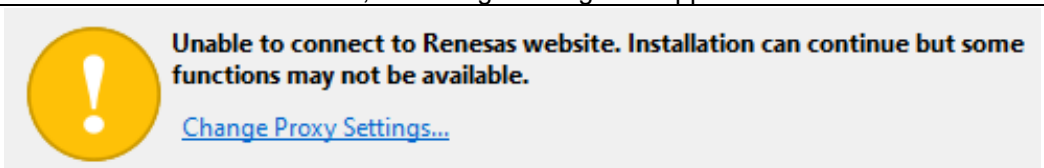


Figure 2-4 e² studio Installation – Website Access Warning Message

As stated in the message, the installation can continue. Although the software listed under "Additional Software" (such as compiler products mentioned later) are not installed, they can be installed separately after e² studio installation.

3. Device Families

Select all Device Families you will use. Click [Next] to continue.

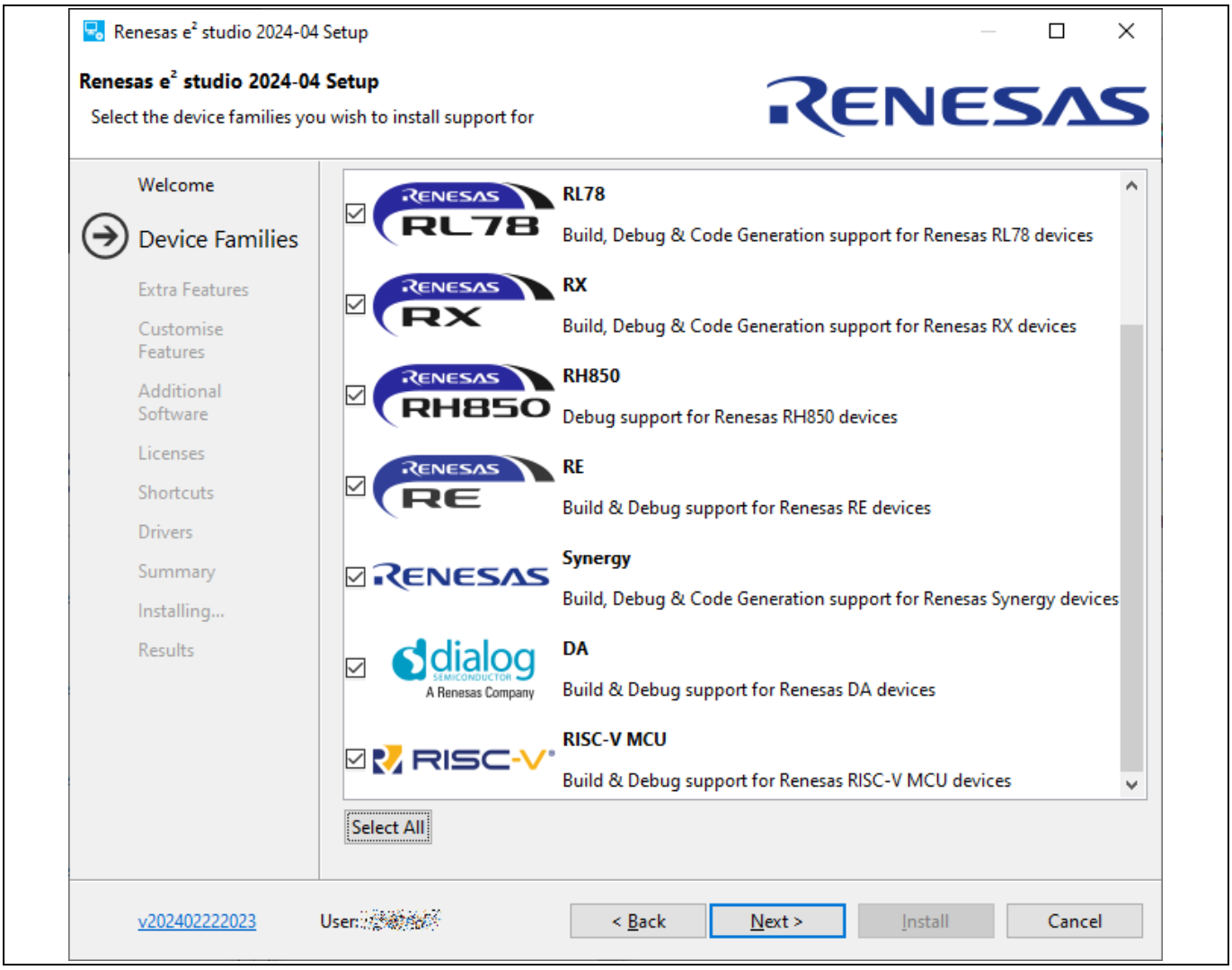


Figure 2-5 e² studio Installation – Device Families

Note: Unchecked device families will not support any build features (project creation, import and build) or debug features. Please ensure that you select the device family that you may possibly use. If you want to add a device family later, rerun the installer and choose the 'Modify' option.

4. Extra Features

This dialog is skipped in Lite/Standard installation type. Select Extra Features (i.e. Language packs, Git support, RTOS support…) to install.
For the non-English language menu, please select Language packs at this step.

Click [Next] to continue.

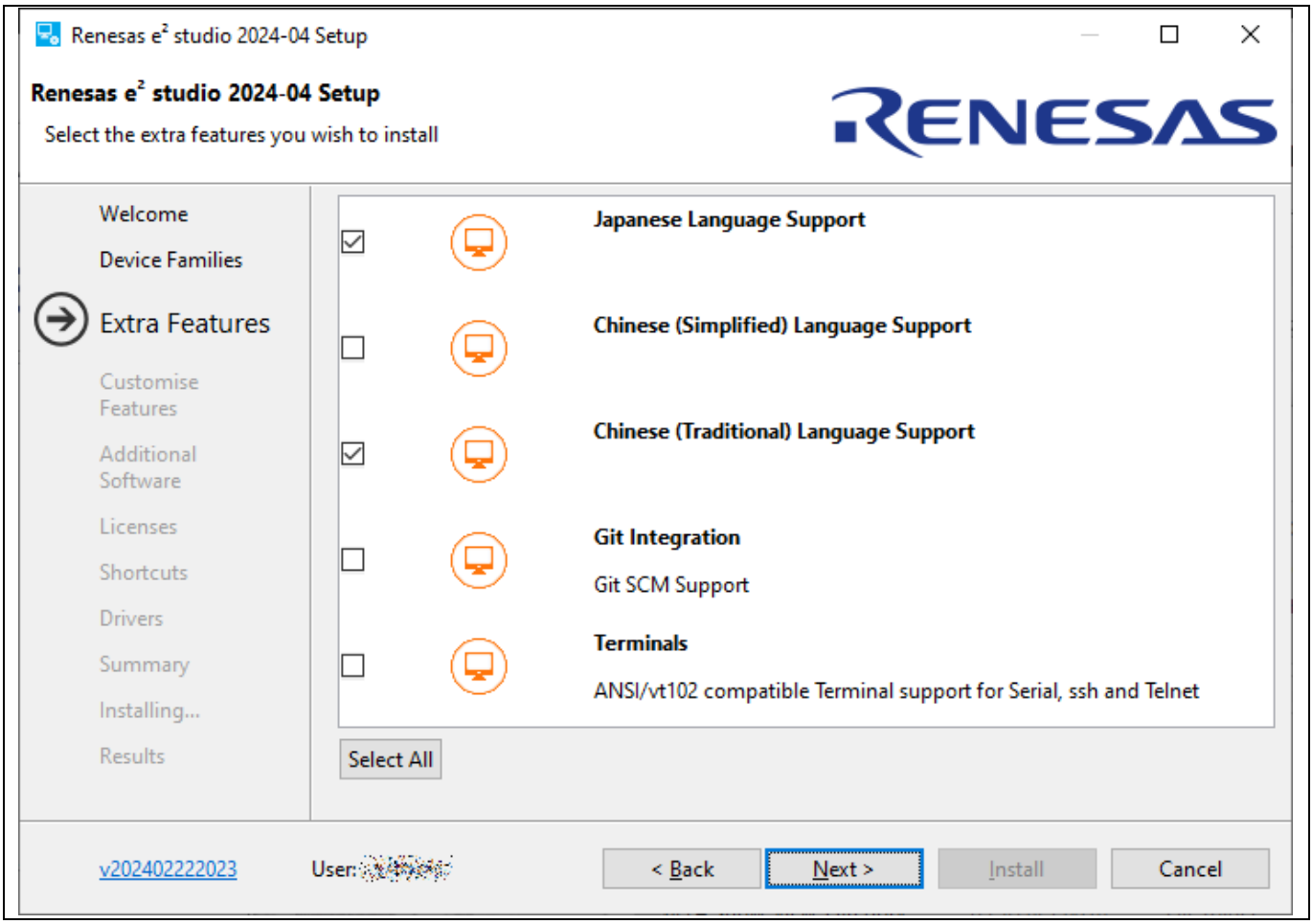


Figure 2-6 e² studio Installation – Extra Features

5. Customize Features

   This dialog is also skipped in Lite/Standard installation type. Select the components to install and click the [Next] button to continue.

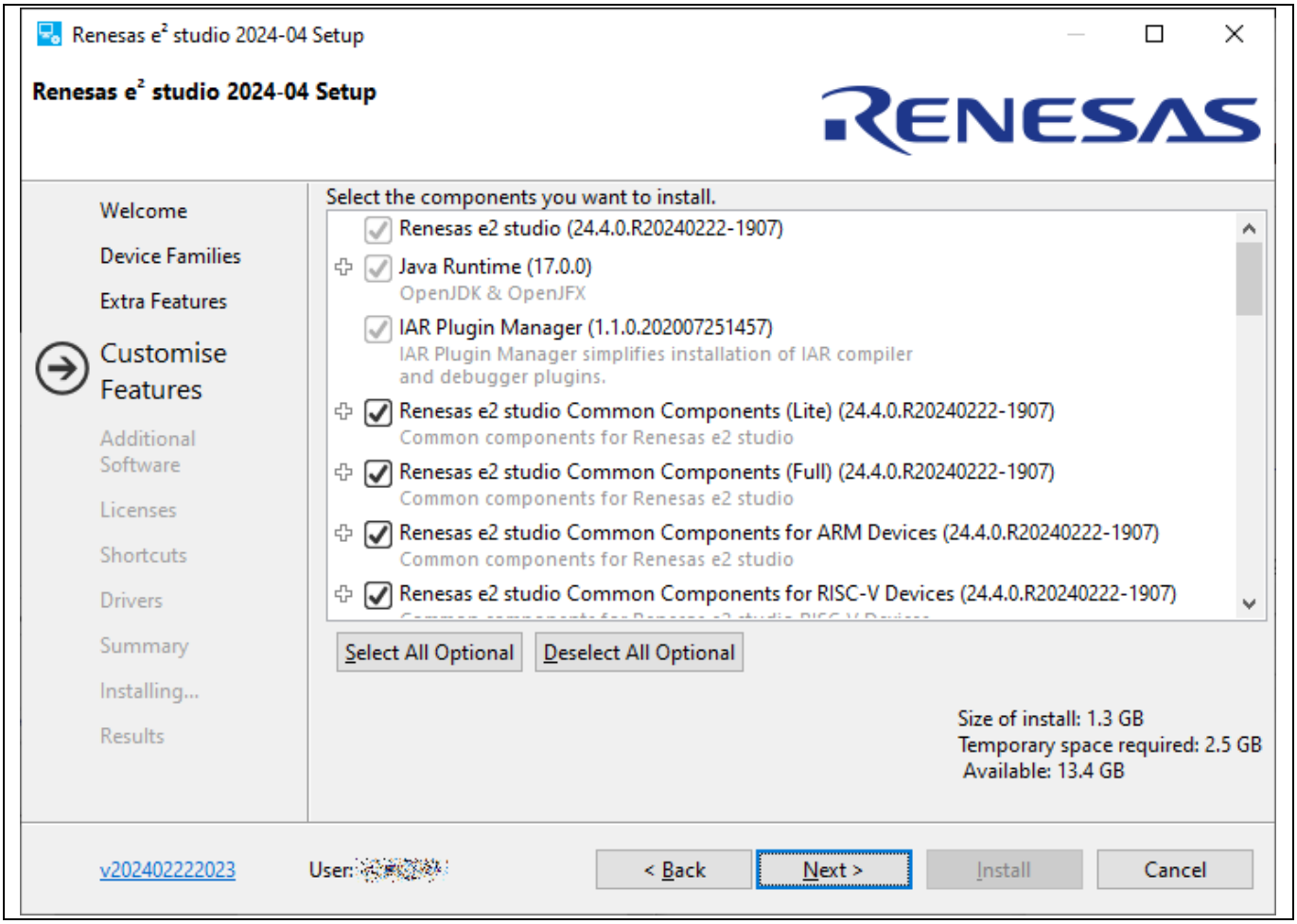


Figure 2-7 e² studio Installation – Customize Features

6. Additional Software

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

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

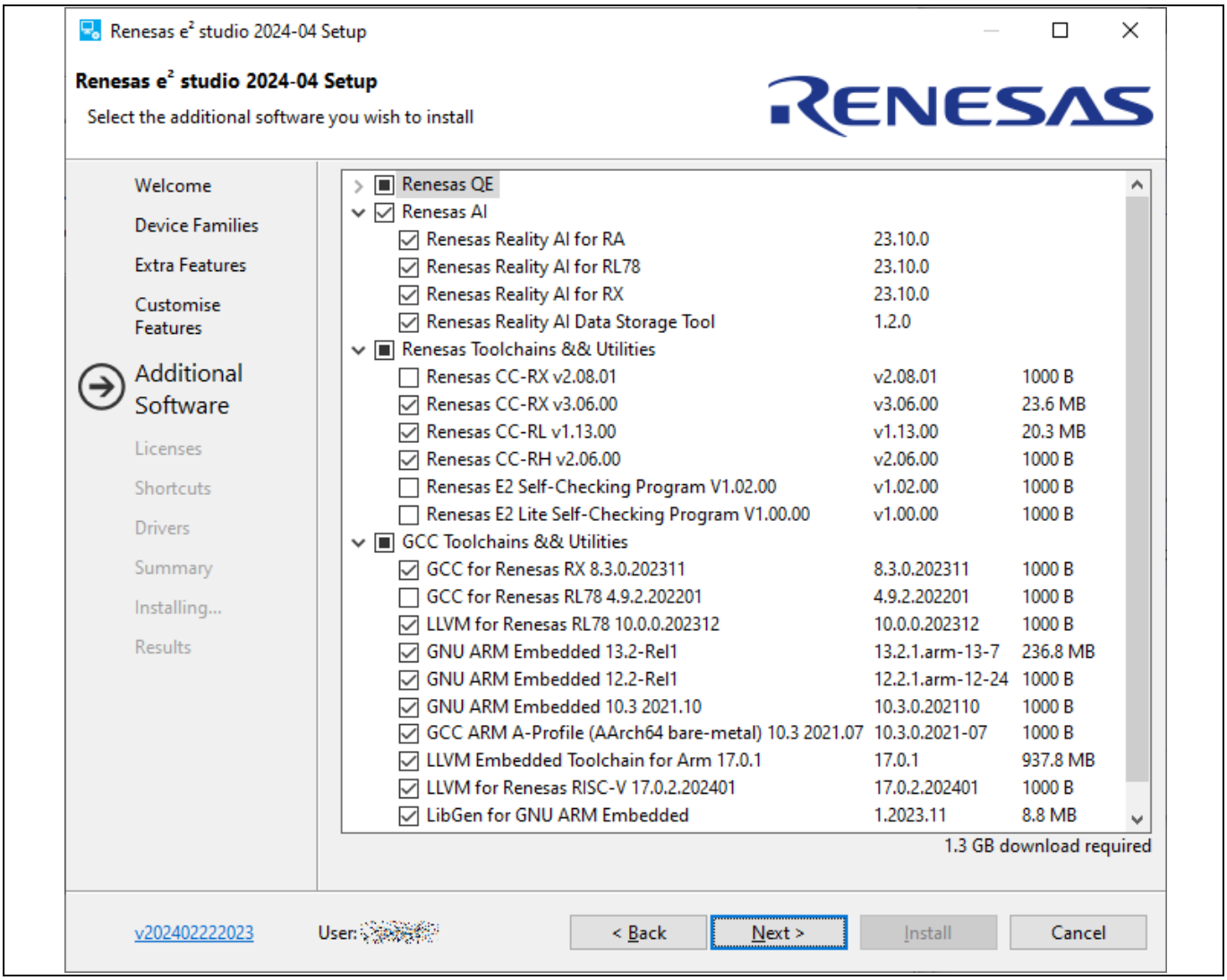


Figure 2-8 e² studio Installation – Additional Software

7. License Agreement

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

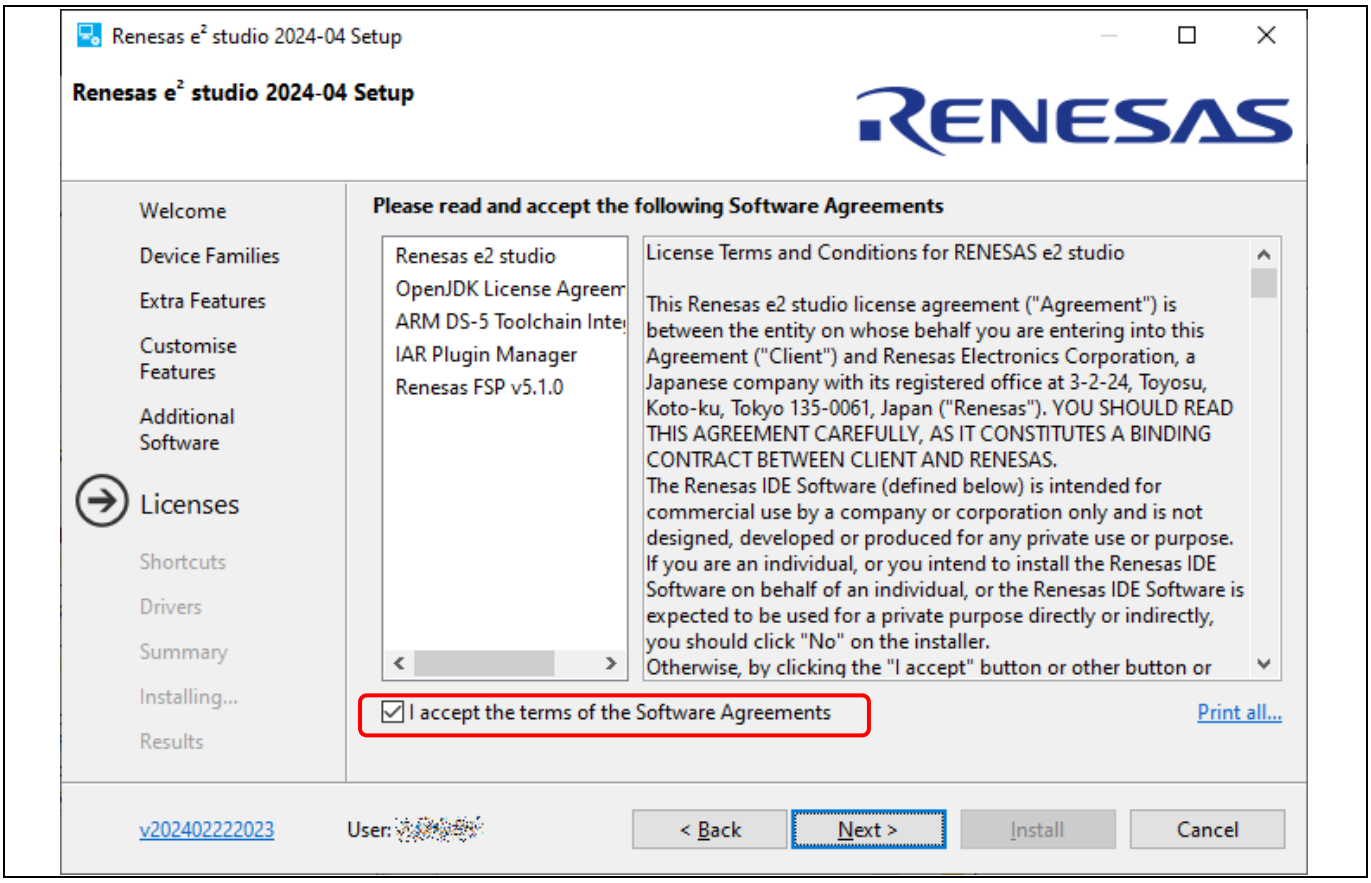


Figure 2-9 e² studio Installation – Licenses

You can also check the contents displayed here after installation. You can read them by clicking the icon in "About e² studio" under the e² studio "Help" menu and select the License button, or by opening "Features" tab in "Installation Details".

8. Shortcuts

Select shortcut name (not mandatory) for start menu and click the [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).

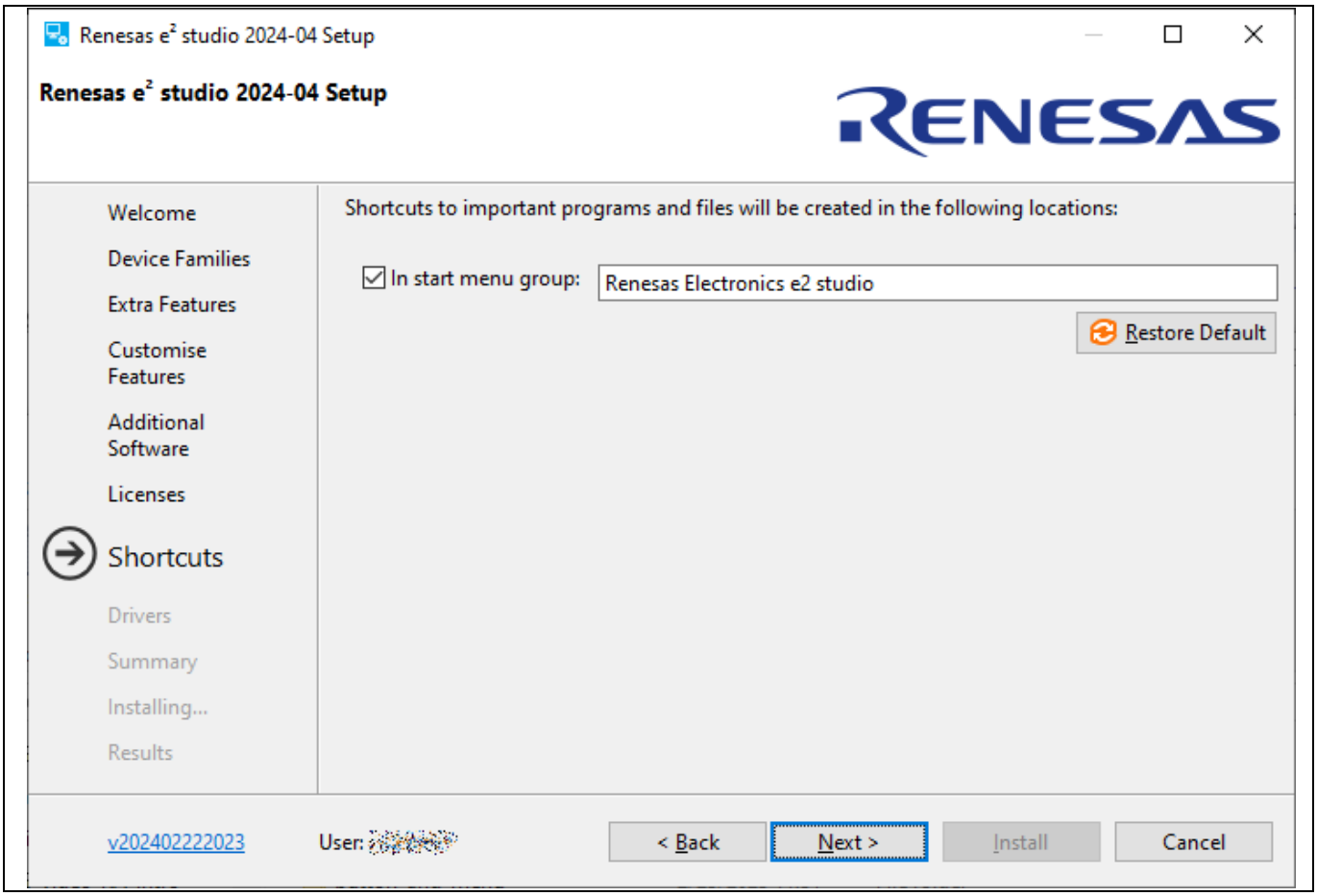


Figure 2-10 e² studio Installation – Shortcuts

9. Drivers

This dialog asks whether to install the debugger driver. Do not install drivers if you have previously installed the same version of the e² studio.

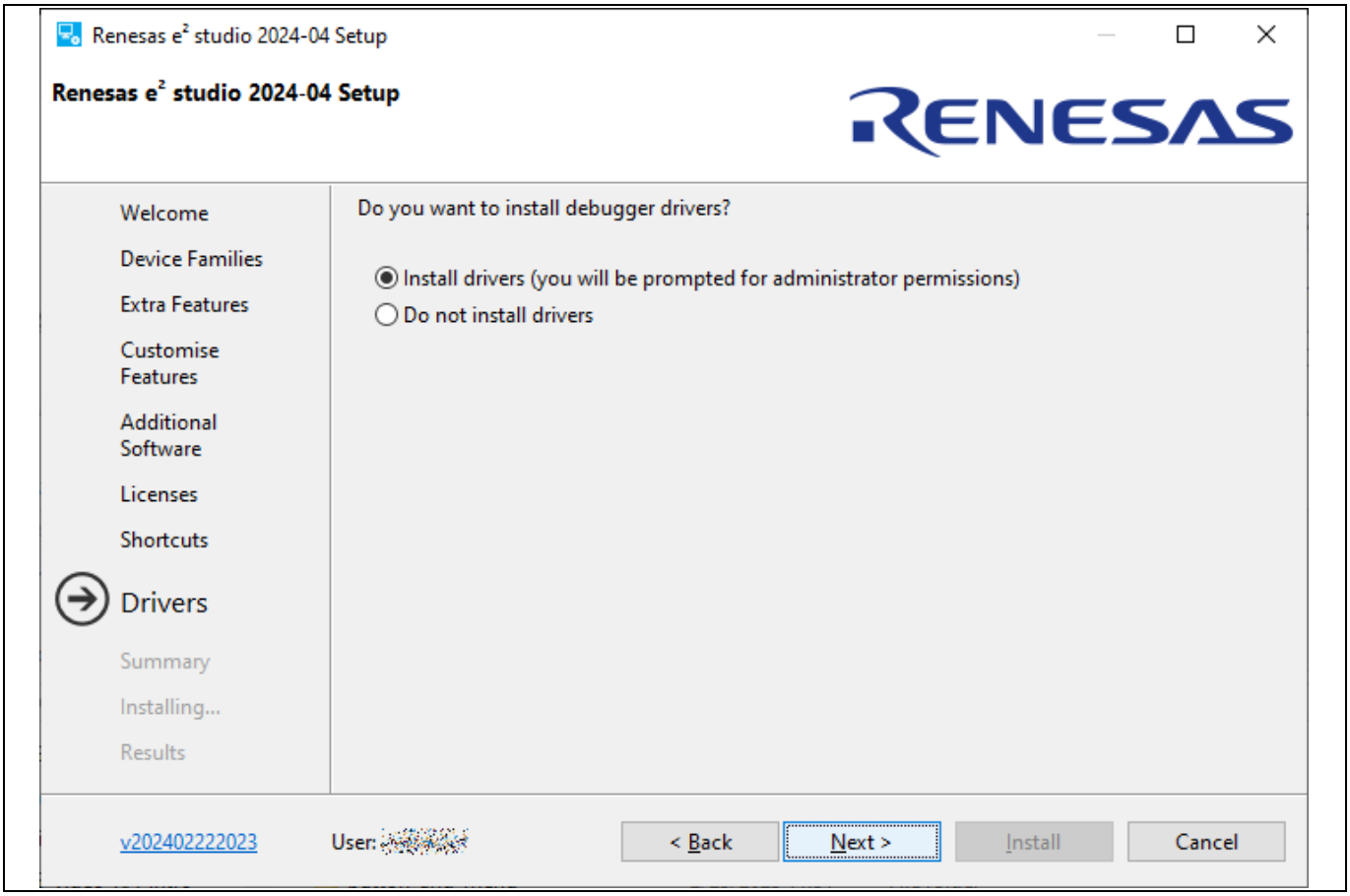


Figure 2-11 e² studio Installation – Drivers

10. Summary

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

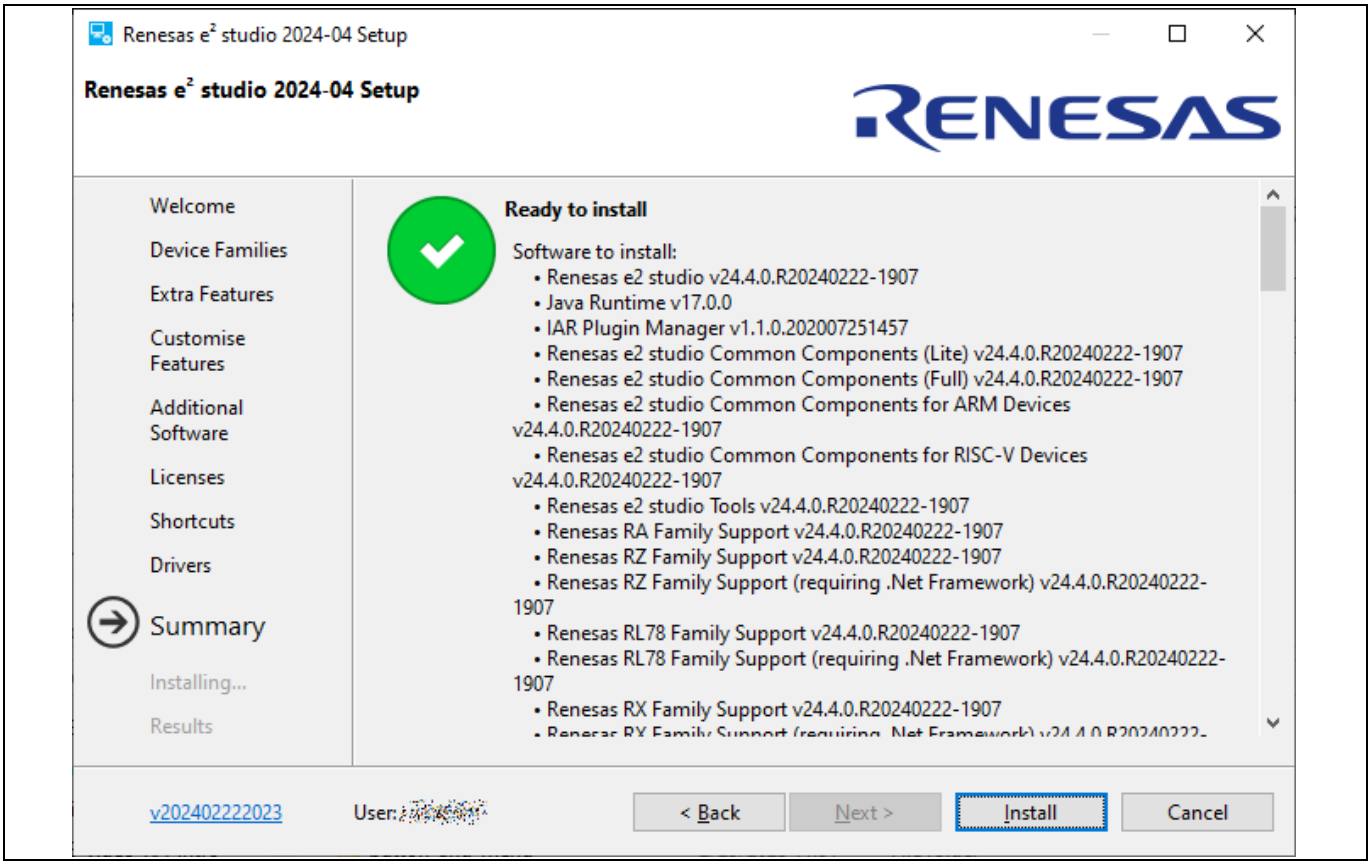


Figure 2-12 e² studio Installation – Summary

11. Installing…

The installation is in progress. Based on selected items of Additional Software, new dialogs are opened to proceed with the software installation.

12. Results

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

Click the [OK] button to complete the installation.

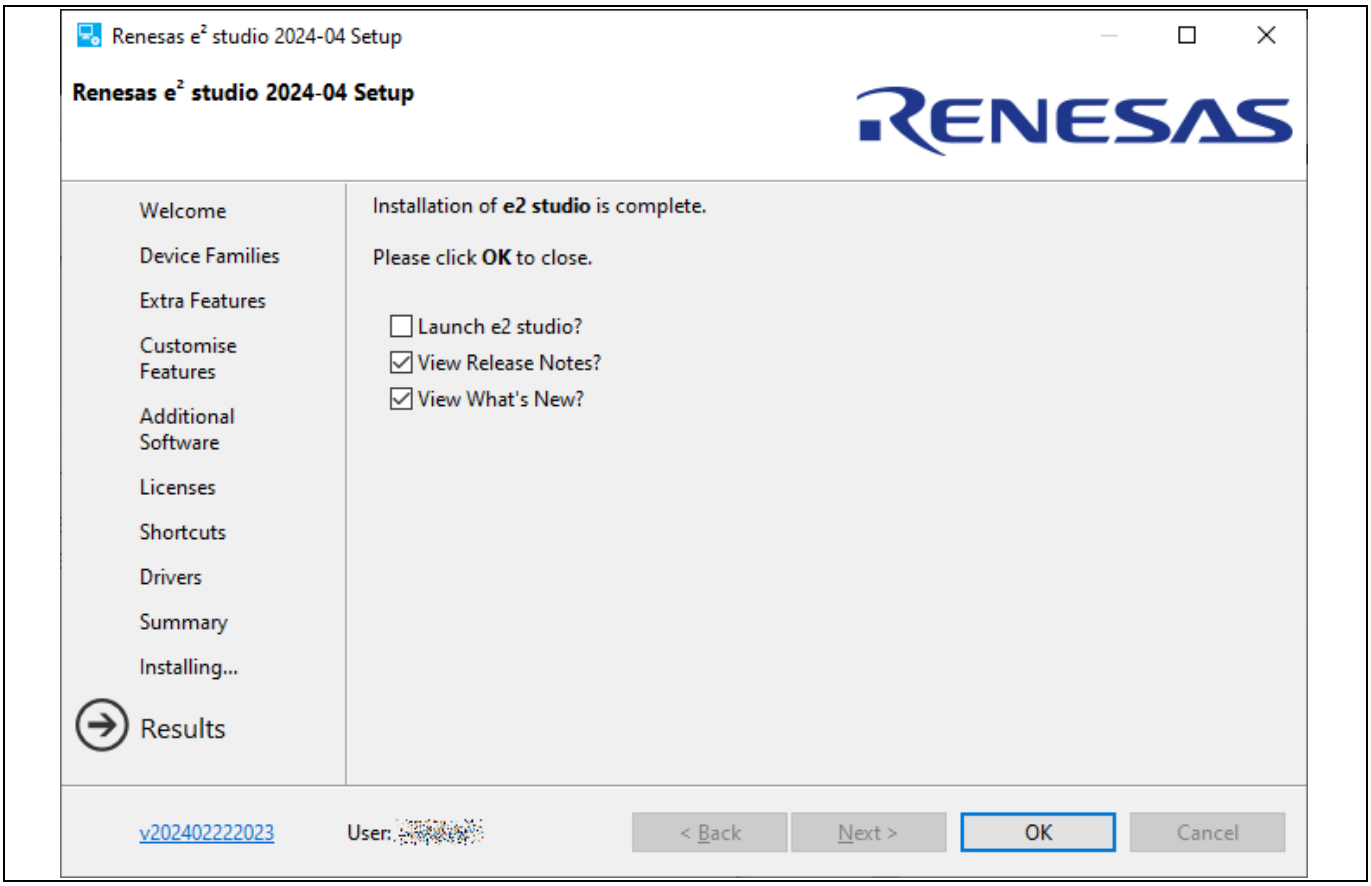


Figure 2-13 e² studio Installation – Results

Note: If you have installed as a different account than the one you normally use such as for administrator privileges, do not start the e² studio by checking "Launch e2 studio?" in this dialog. If you do so, discard the workspace that was used at that time.

2.2 About 32-bit Version of e² studio

There is no Lite/Standard selection for e² studio V7.8.0 or earlier installers.

The installer's interface follows a similar flow to that of the 64-bit version, although the displayed content may vary slightly. Therefore, refer to the above explanation when using it.

For users of the 32-bit OS environment in which 64-bit versions of e² studio do not work, e² studio V7.8.0 is available on the e² studio product page. Unless there is a specific requirement for this version, it is recommended to use the 64-bit versions (202x-xx).

2.3 Uninstallation of the e² studio IDE

By Windows OS, you can uninstall programs by selecting "e² studio" from the list of installed programs in [Start] → [Control Panel] → [Apps & features]. However, if uninstallation is not possible, run the uninstaller directly from the following folder:

{e² studio installed folder}/uninstall/**uninstall.exe**.

If this still doesn't work, remove it using the same version of the e² studio installer, or just delete the destination folder.

2.4 Installation of Compiler Packages

As mentioned above in installation procedure, compiler packages can be installed with e² studio when you have access to the Internet. Please install compiler packages separately from e² studio installation when Internet access is not available.

Compiler packages are found in the following websites. Please read descriptions on the websites for installation procedures.

Renesas Compiler for RX : http://www.renesas.com/rx_c

Renesas Compiler for RL78 : http://www.renesas.com/rl78_c

Renesas Compiler for RH850 : http://www.renesas.com/rh850_c

LLVM and GNU toolchain for RL78 : https://llvm-gcc-renesas.com/rl78-download-toolchains/

GCC for Renesas RX toolchain: https://llvm-gcc-renesas.com/rx-download-toolchains/

LLVM for Renesas RISC-V MCU toolchain : https://llvm-gcc-renesas.com/riscv-download-toolchains/

Open “Renesas Toolchain Management” dialog ([Help] menu -> [Add Renesas Toolchains]) to confirm which toolchains are installed and integrated. The checked toolchains are integrated into e² studio. Click the [Add] button to add a new toolchain installed path if it is not found in this list.

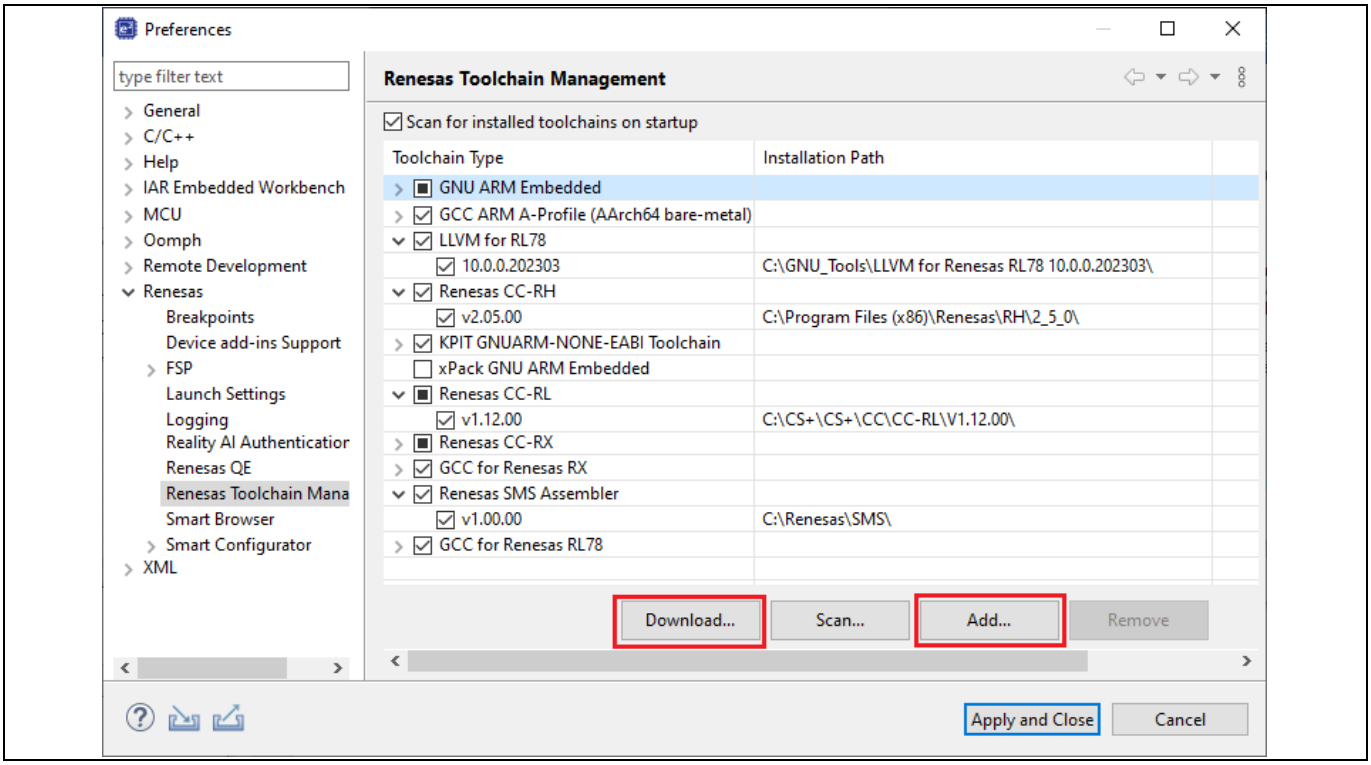


Figure 2-14 Renesas Toolchain Management

You can also click the [Download] button to install a new toolchain when you have Internet connection.

2.5 Tutorial

The following chapters will explain the process of creating a project to start the debugger. Additionally, you will also find a tutorial for each type of Device Family and toolchain (compilers) in the e² studio Help.
Note: Tutorials are available for the supported Device Families that you selected during installation.

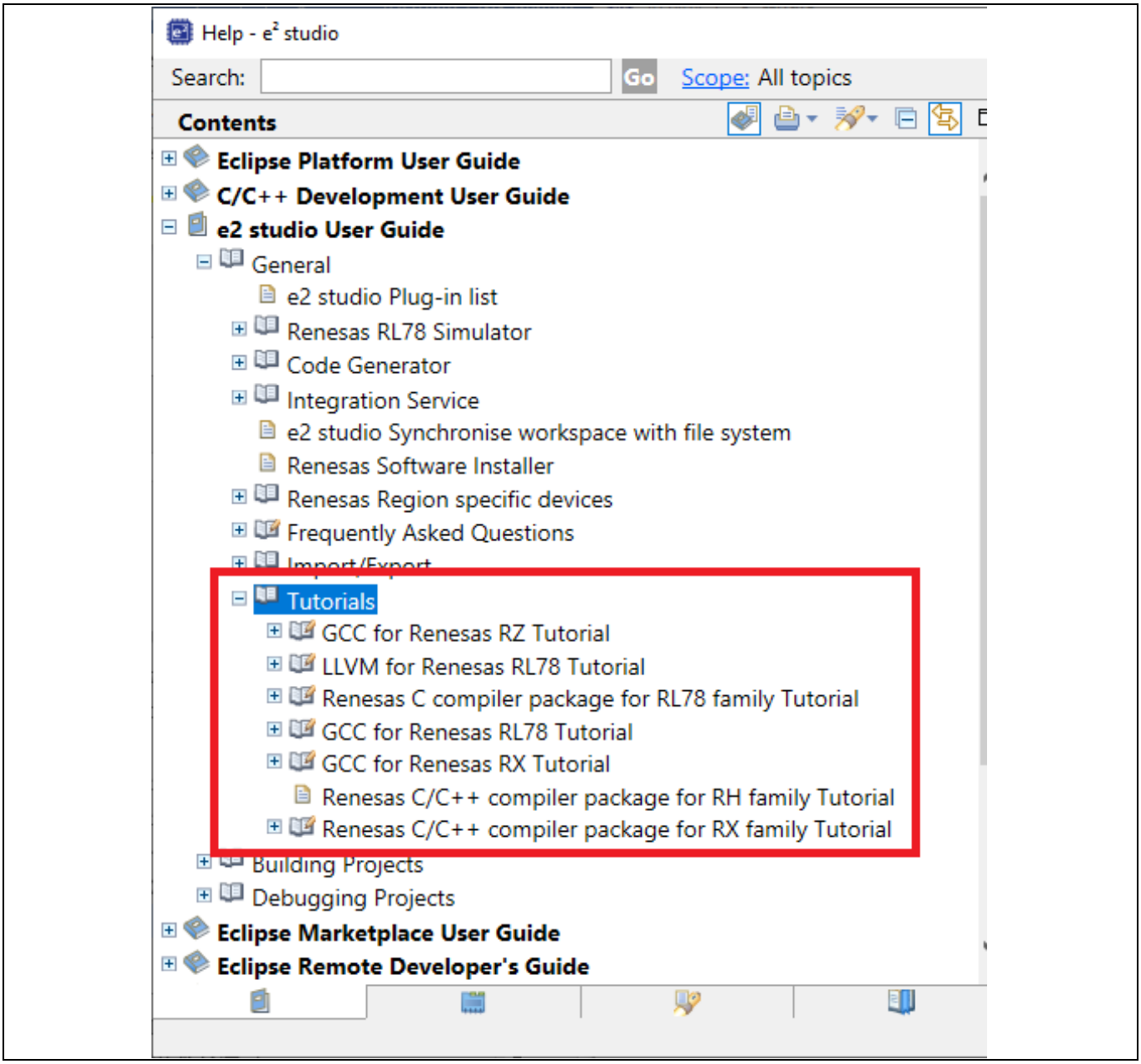


Figure 2-15 Tutorial

3. Project Generation

In e² studio, "Project" is a basic unit to perform build and debug operations.
This chapter describes the creation of a new project and import of an existing e² studio project, the High-performance Embedded Workshop IDE (described as “HEW” below) project and the CS+ project to the 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 the 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 the e² studio IDE from Windows ([Start] menu) and specify a workspace directory. To create a new project, proceed as follows:

Note: For DA devices, do not create a new project but import a project by following section 3.5, Importing SDK Projects for DA Devices.

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

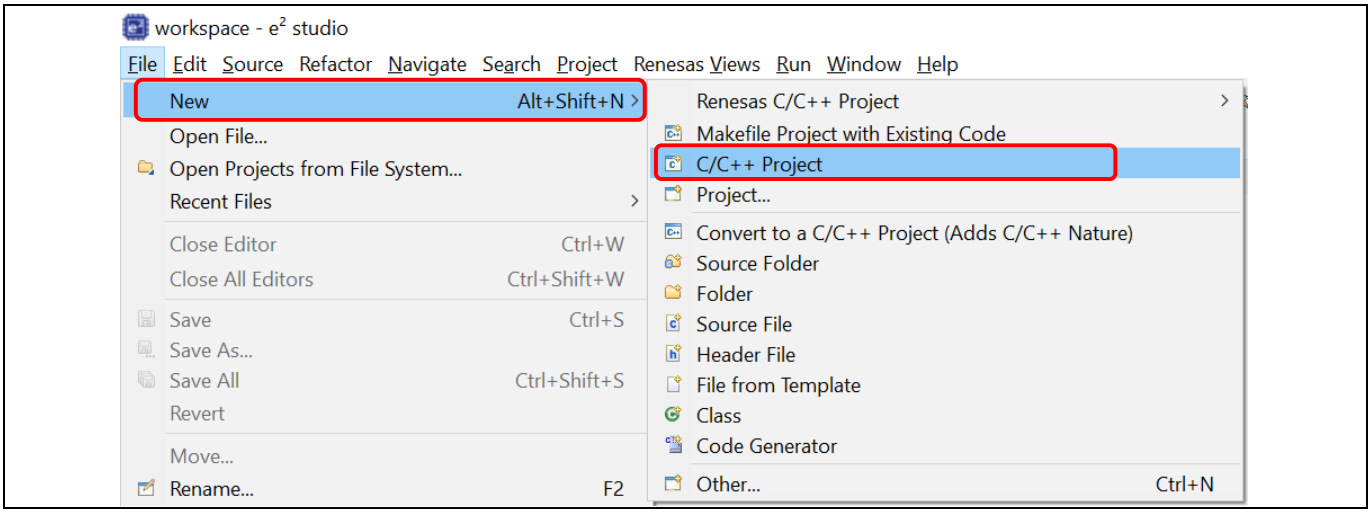


Figure 3-1 Open New Project Creation Wizard

2. Select a template for the new project (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 the e² studio installer to add "Build/Debug support plugins" for the target device family. Click [Next] to proceed.

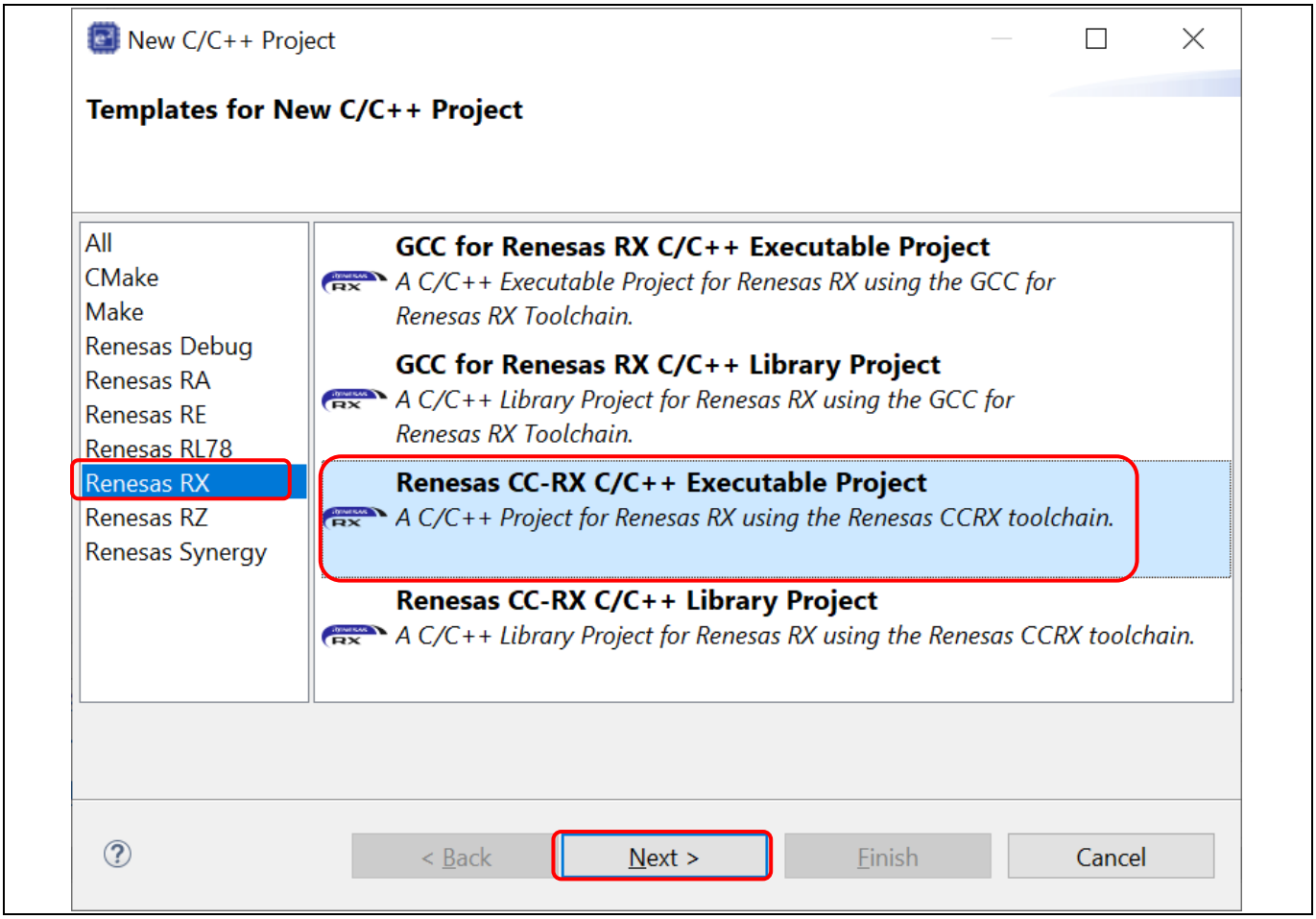


Figure 3-2 New Project Creation Wizard (1/6)

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

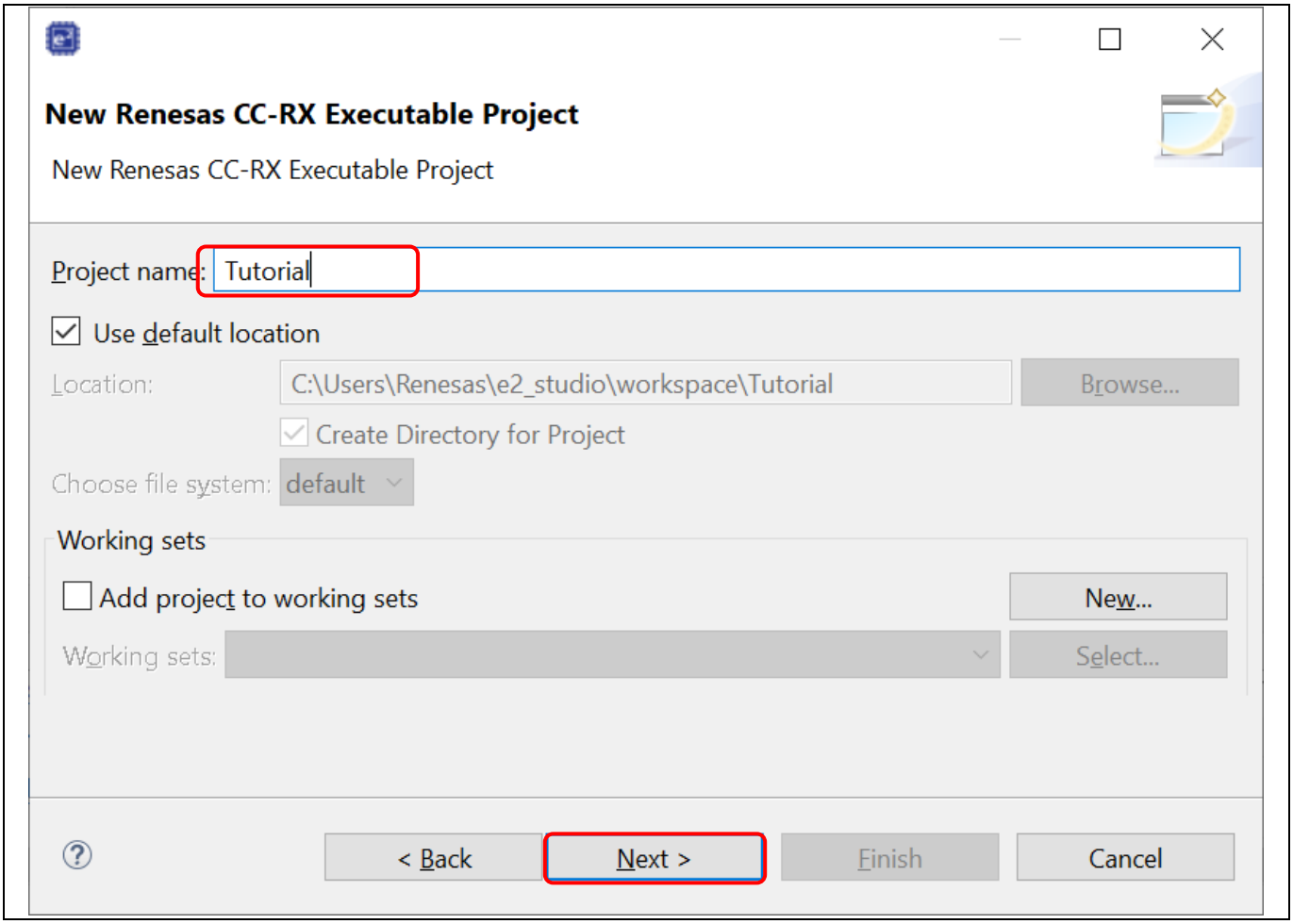


Figure 3-3 New Project Creation Wizard (2/6)

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

   Notes:

   - "E1" or “E2” can be selected in the same way as E2 Lite in the Hardware Debug Configuration pull down menu.
   - Please select “C++” as Language when you use C++ source files in the project.

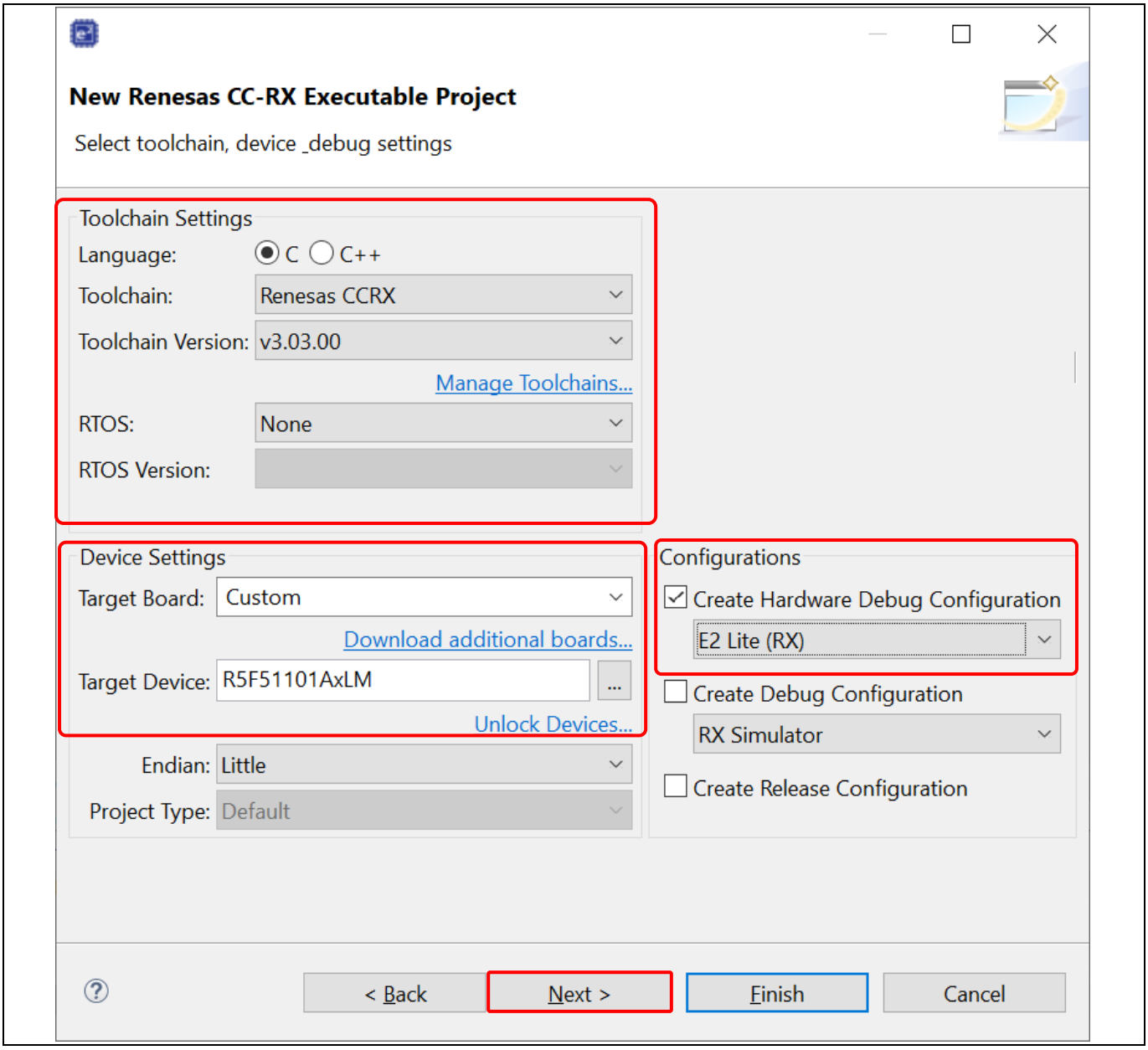


Figure 3-4 New Project Creation Wizard (3/6)

5. Coding Assistant feature can be applied if necessary. Click [Next] to proceed.
   - *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 in the higher layer than CG, such as communication protocol stacks 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.

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

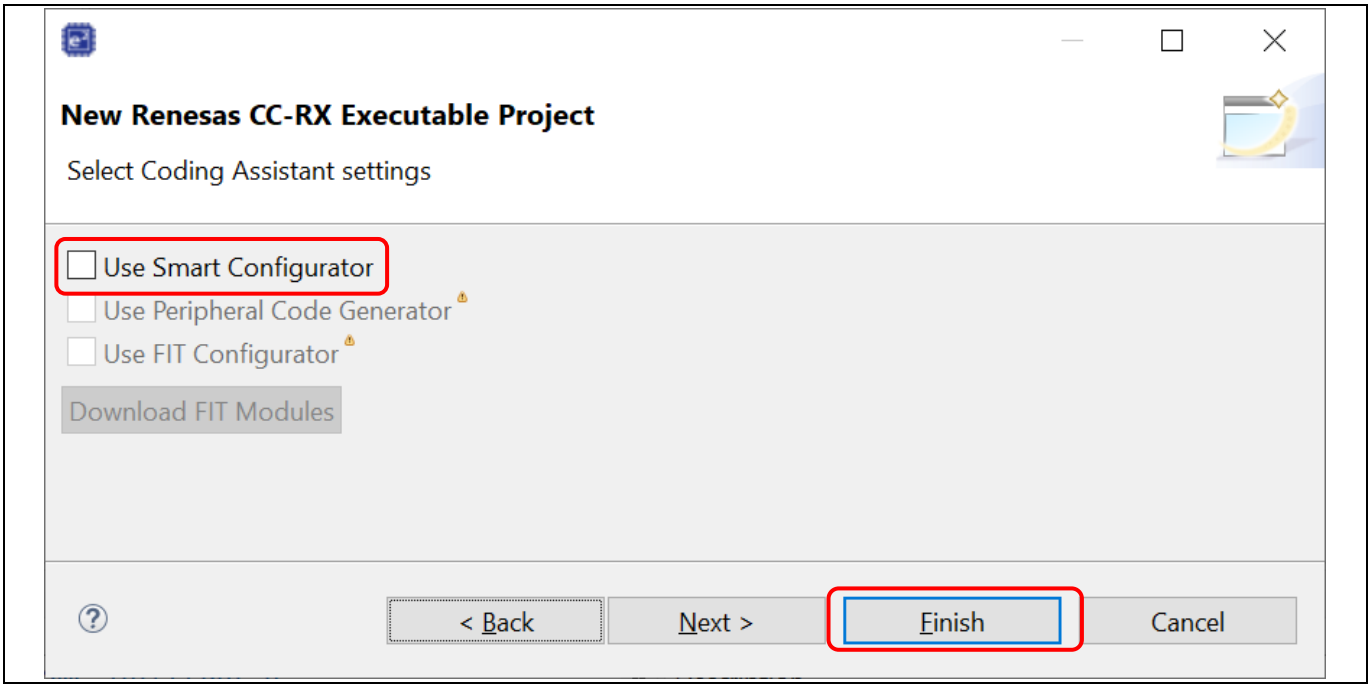


Figure 3-5 New Project Creation Wizard (4/6)

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

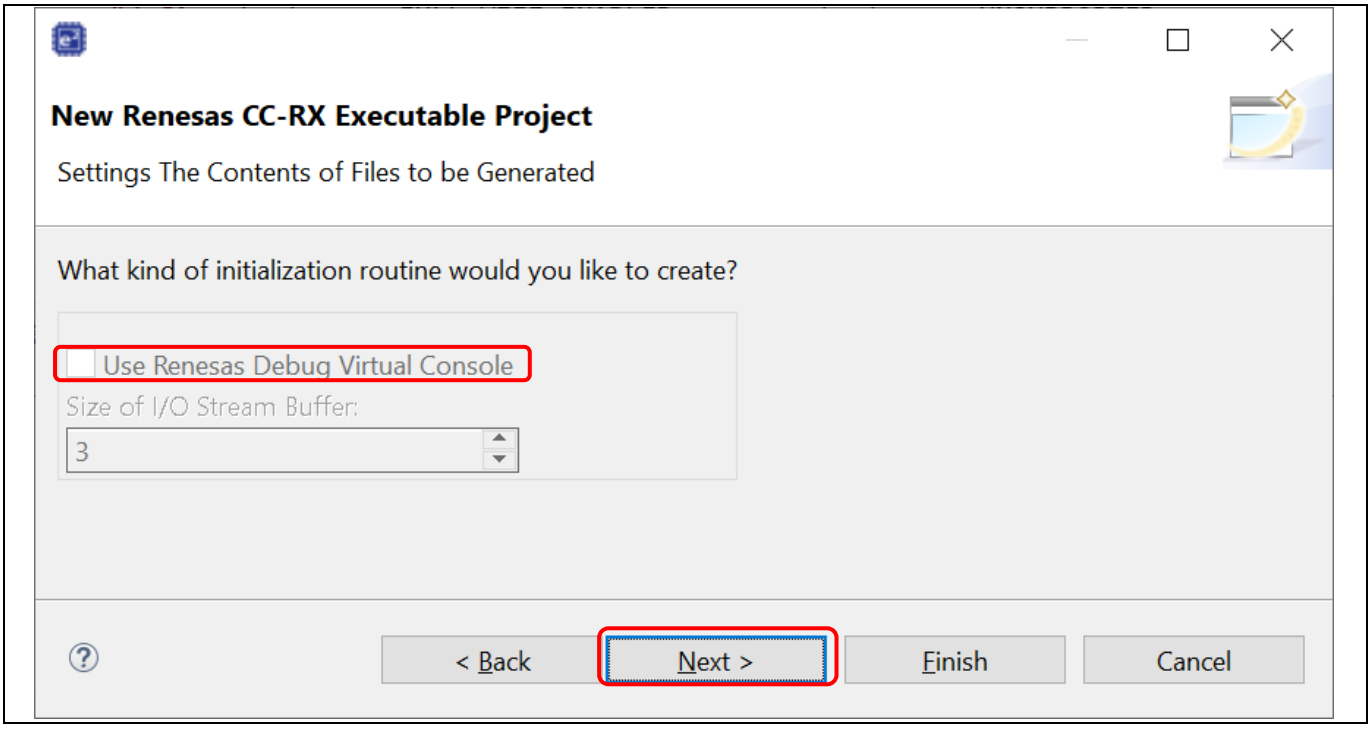


Figure 3-6 New Project Creation Wizard (5/6)

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

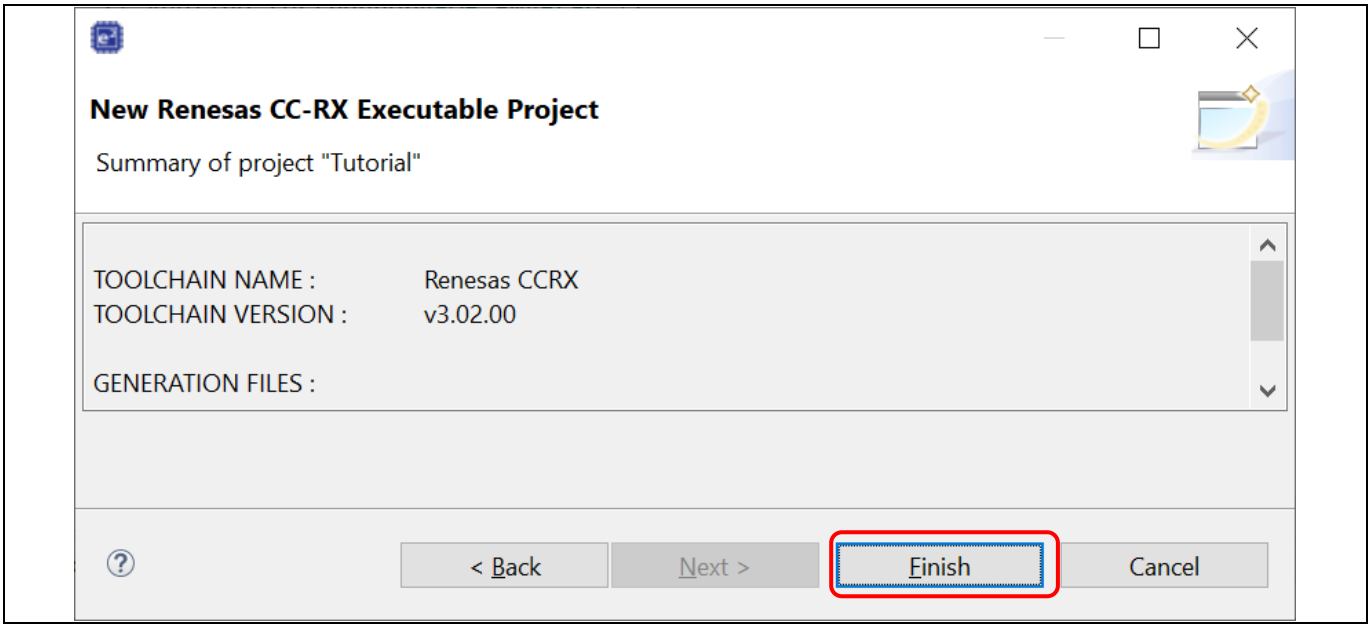


Figure 3-7 New Project Creation Wizard (6/6)

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

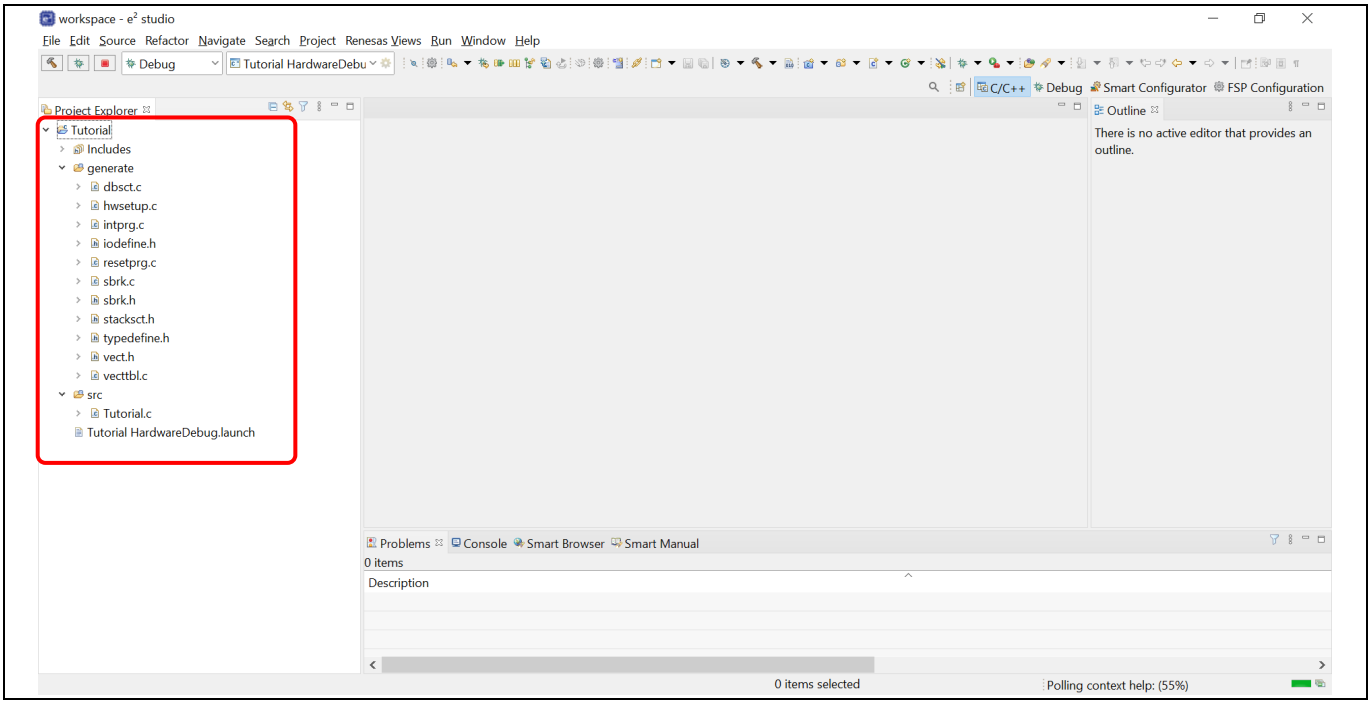


Figure 3-8 New C Project Created

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 projects 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 on backing up projects:

- Project properties are stored in files or folders of 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 purposes.
- In order to restore properties shared among projects, for instance when one project makes reference to another project's files, please back up the whole workspace folder.

3.2 New Debug Only Project Generation

Creating a debug only project allows users to debug an existing executable file that users have already built. This feature will automatically create a project and debug configuration for users.

Note: 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,

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

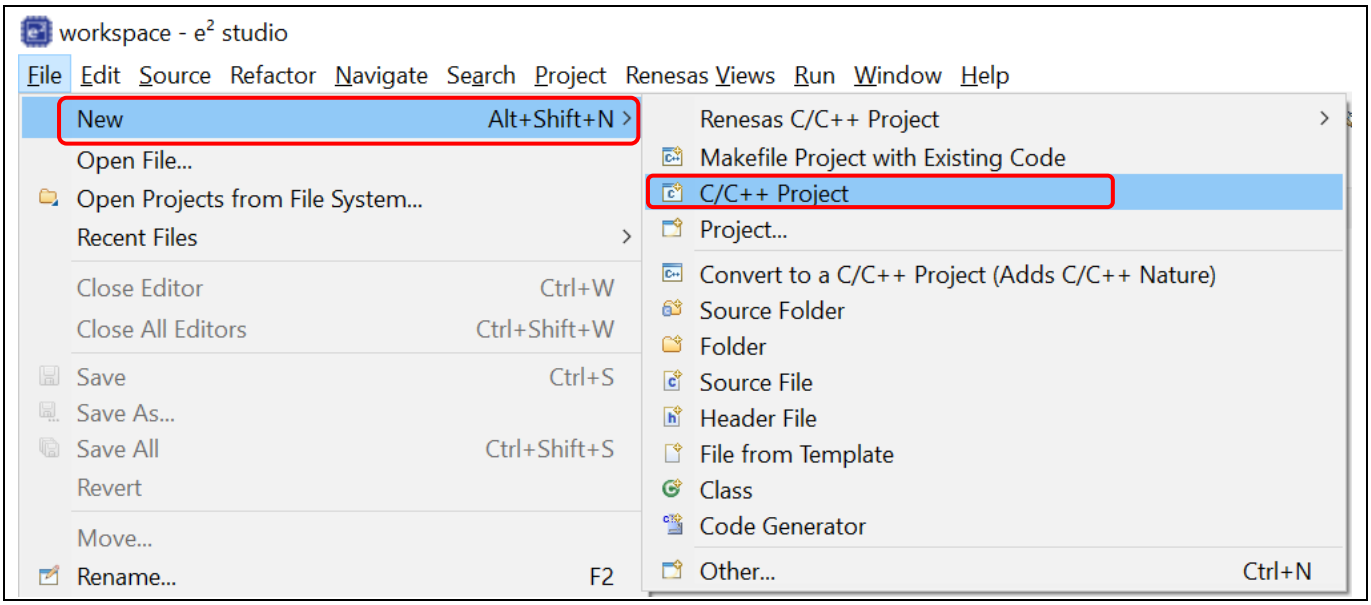


Figure 3-9 Open New Project Creation Wizard

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

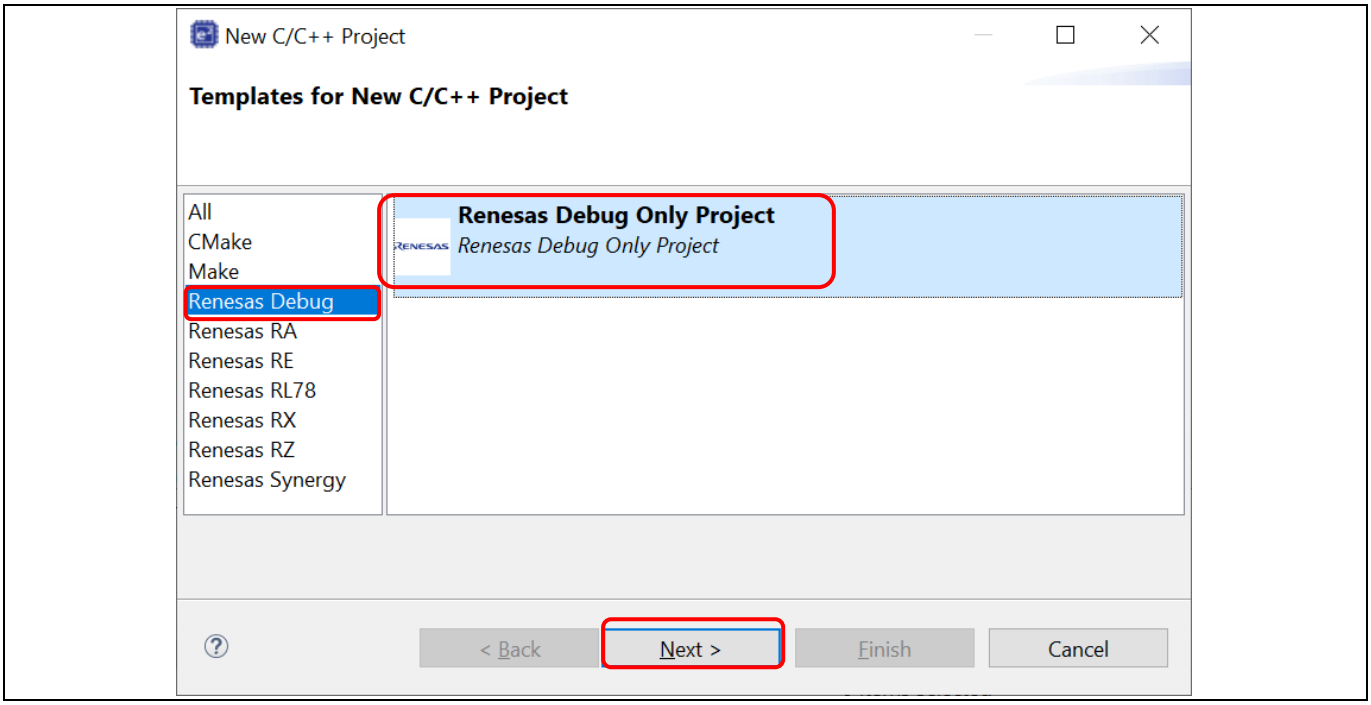


Figure 3-10 Specify the Project Template

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

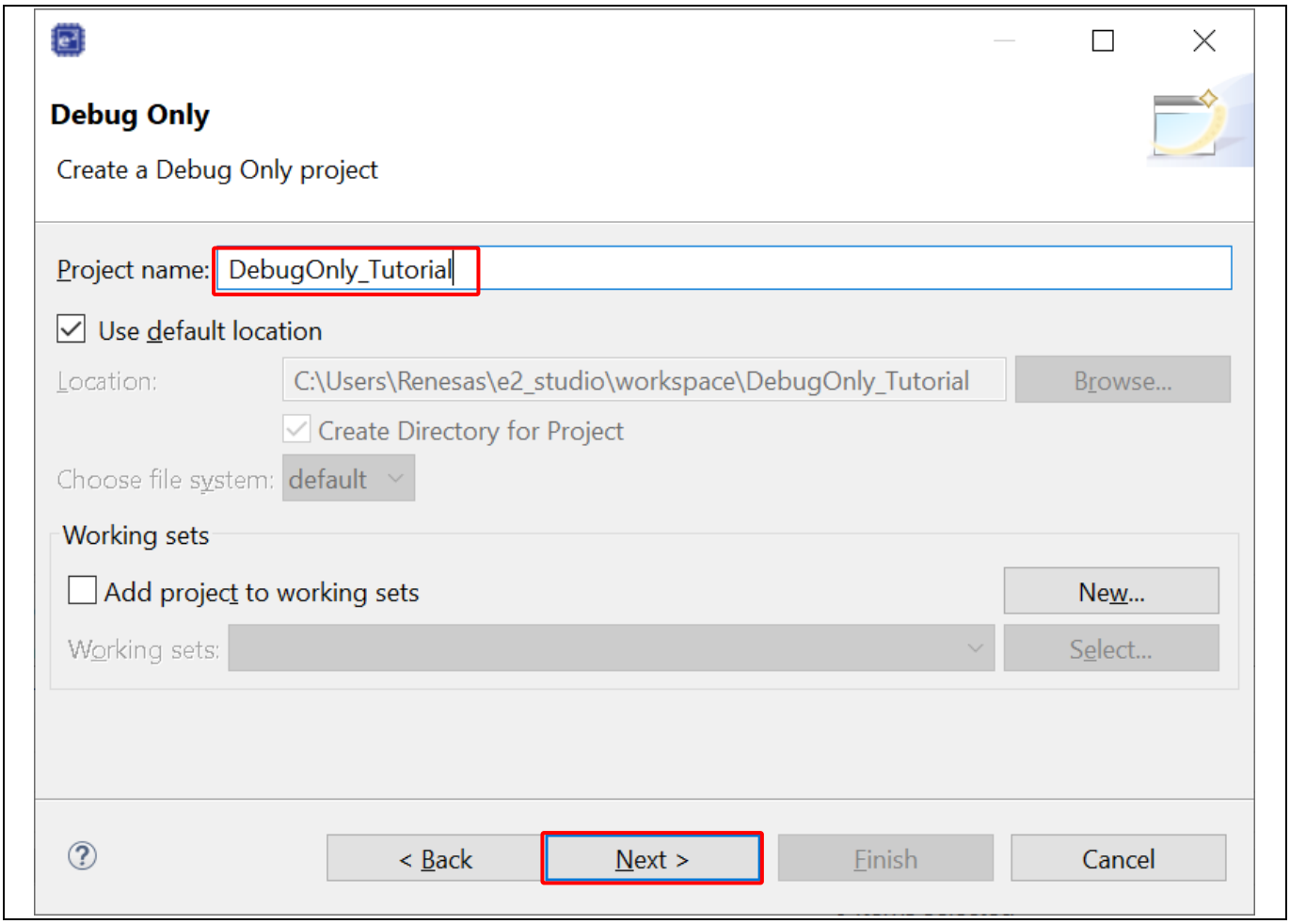


Figure 3-11 Specify Project Name

4. Select the debug hardware (e.g. “E2 (RH850)”) and target device (e.g. R7F701007xAFP). 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 the debugger. Click [Finish] to create the project.

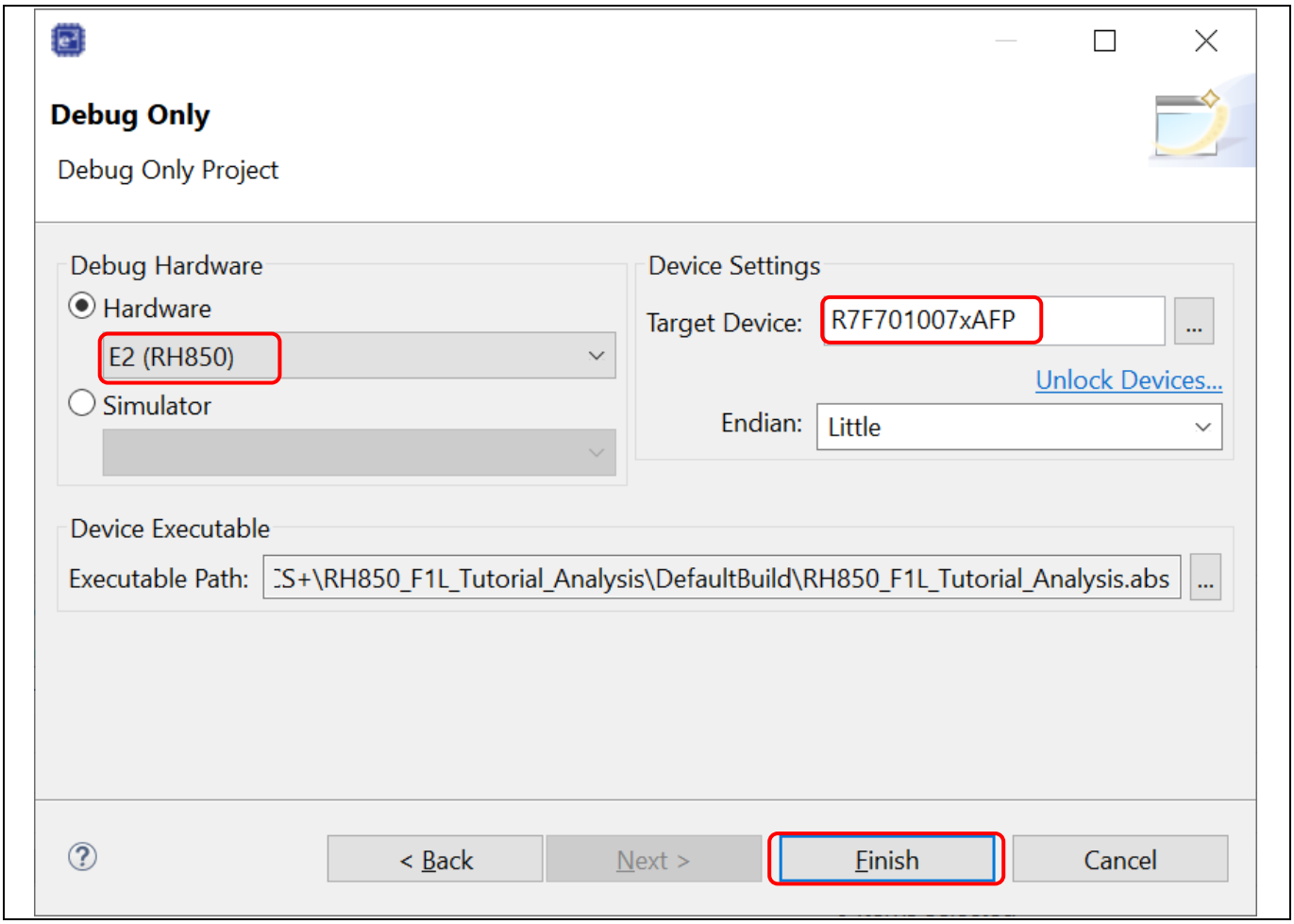


Figure 3-12 Specify Project Settings

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

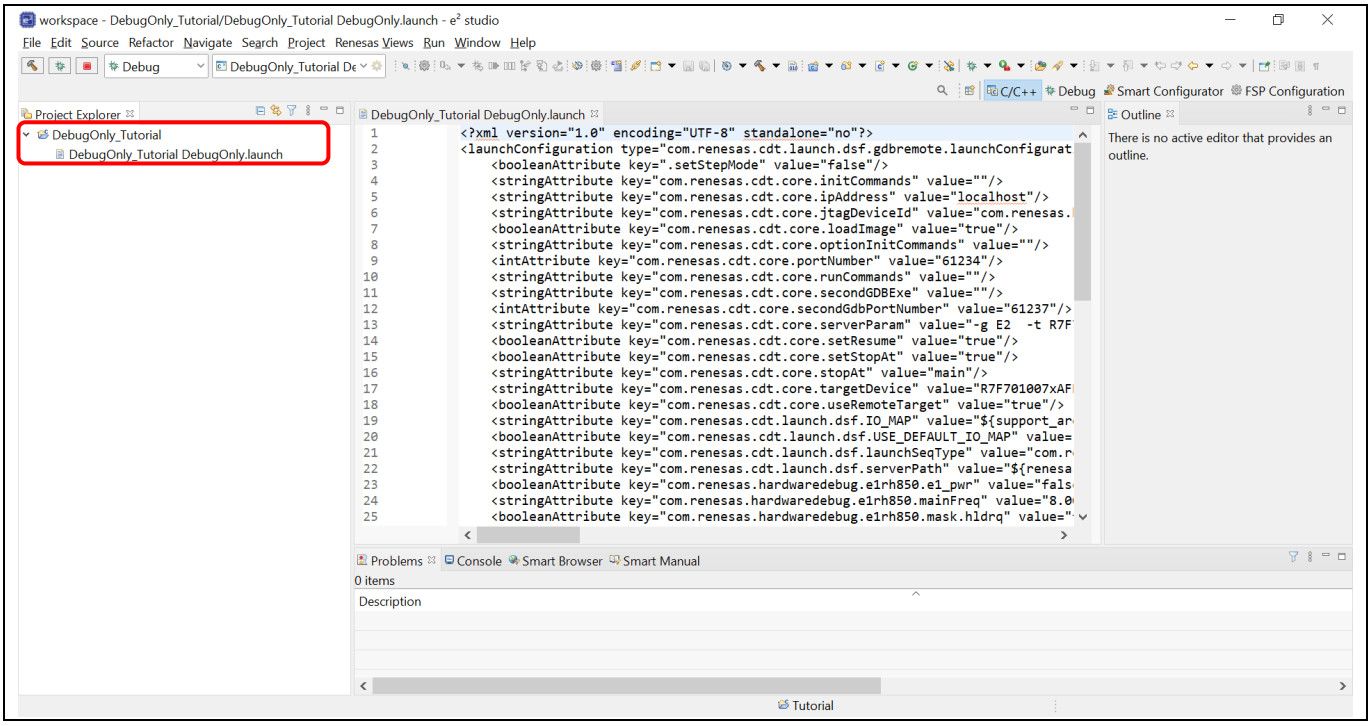


Figure 3-13 Debug Only Project is Created

3.3 Importing a Sample Project

e² studio can search and import sample projects on the Renesas website.
Open the “File” menu and select “Import...” to launch Import Wizard. Then select “Sample Project on Renesas Website”.

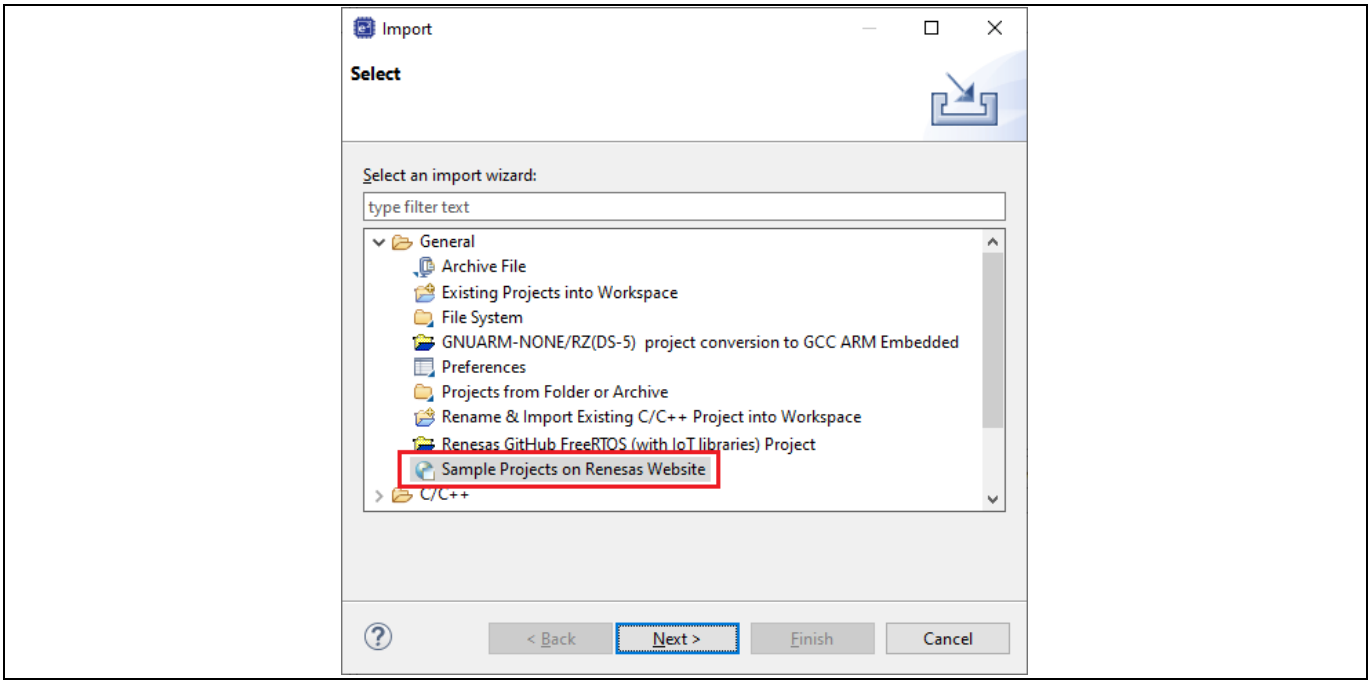


Figure 3-14 Sample Project Import

Select the target device family in the Online or Local category, then select a device. Sample projects will be listed for the device. Click [Finish] to import the selected project. If the project does not appear on this dialog due to PC or networking limitations, refer to the alternative method mentioned in the following chapter.

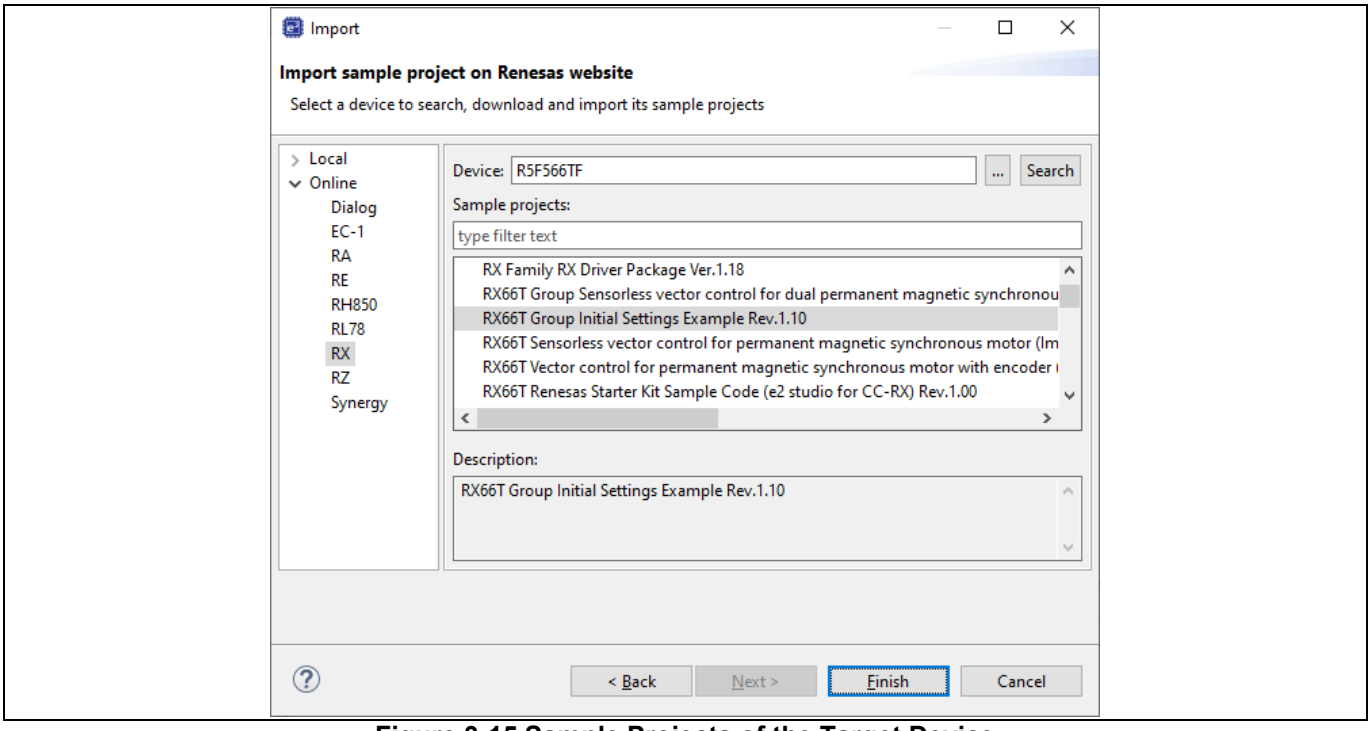


Figure 3-15 Sample Projects of the Target Device

3.4 Importing Projects into the 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/us/en/software-tool/migration-tools-ide

3.4.1 Import Existing Projects

To import an existing e² studio project to a current workspace, please follow instructions below. These steps import a sample project from the Renesas website to use for demonstrating debugging features in section 5.

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

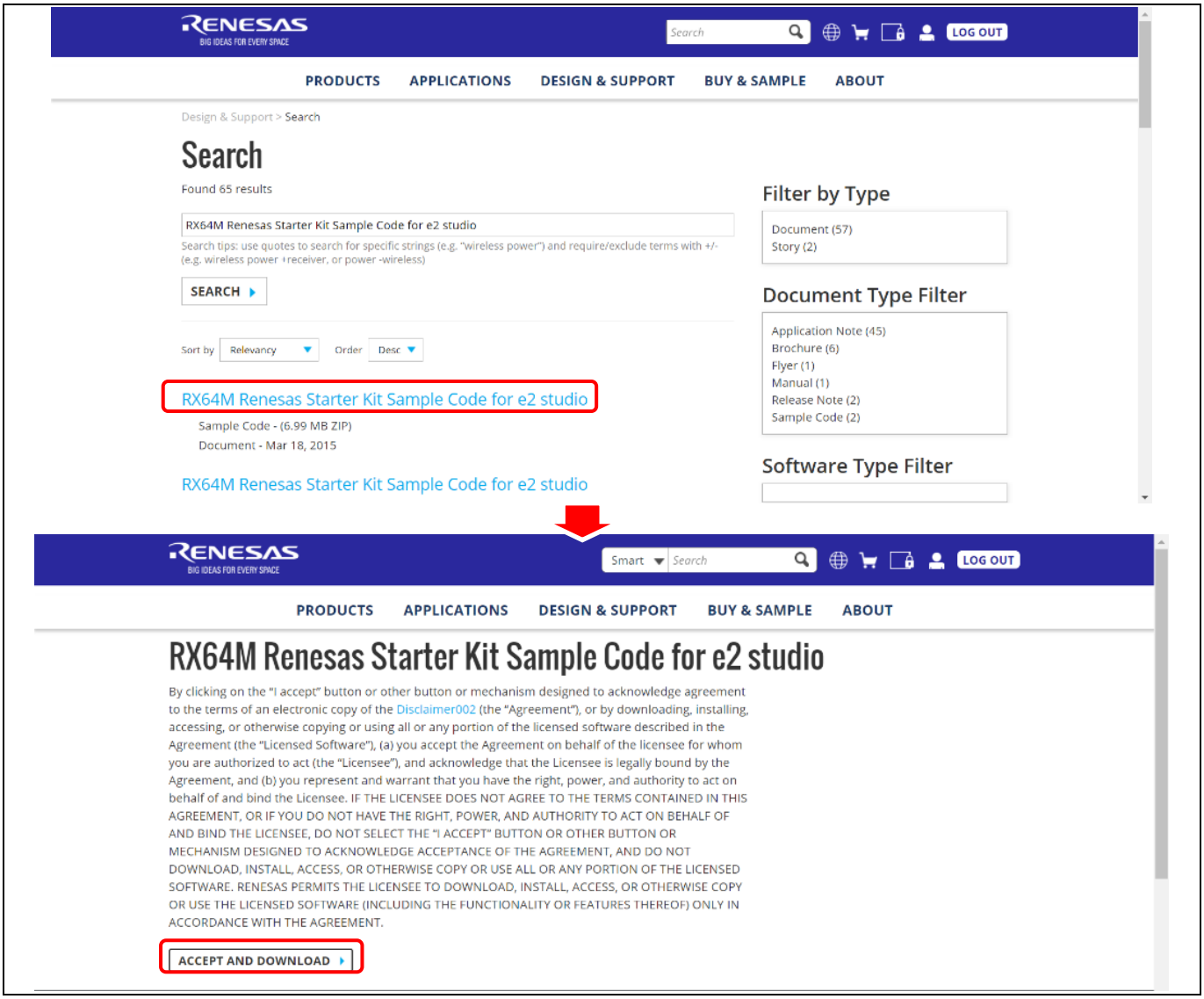


Figure 3-16 Download the Sample Code

2. “Tutorial” project is included in the download file.

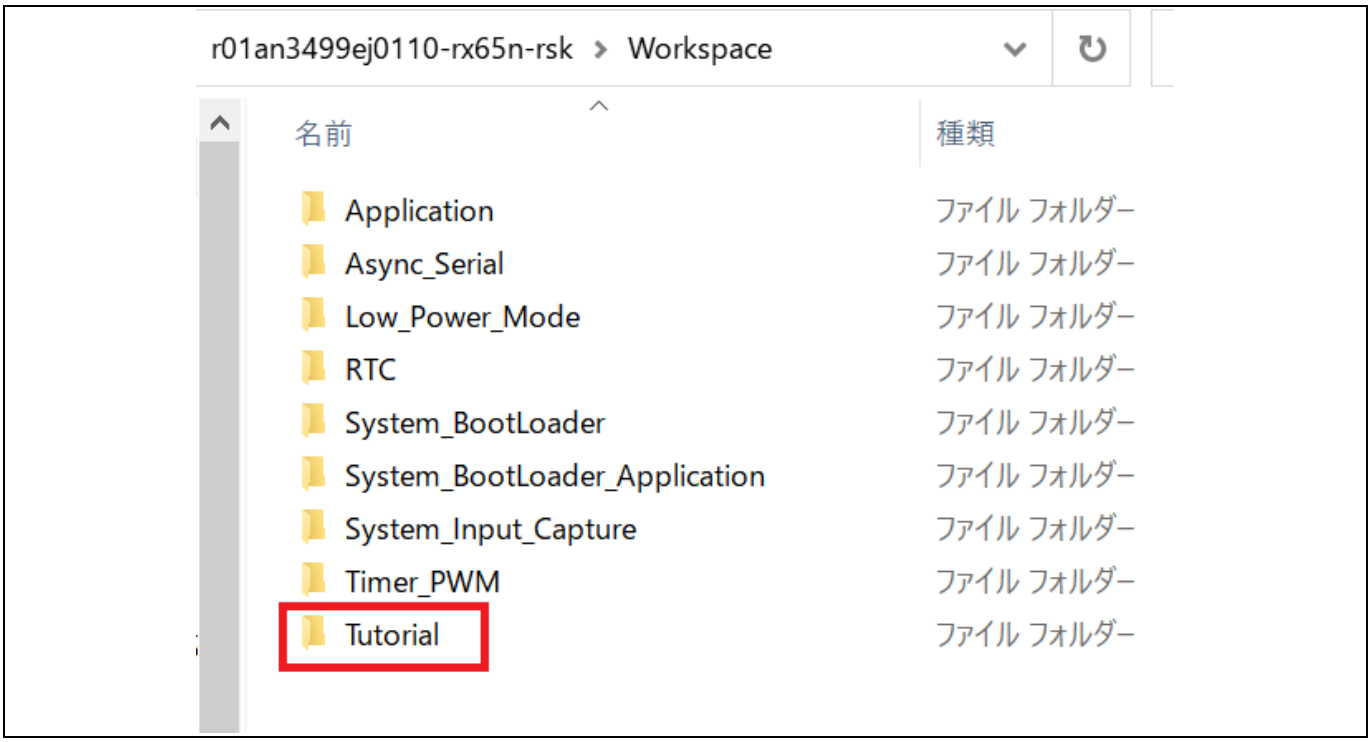


Figure 3-17 Downloaded File Contents

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

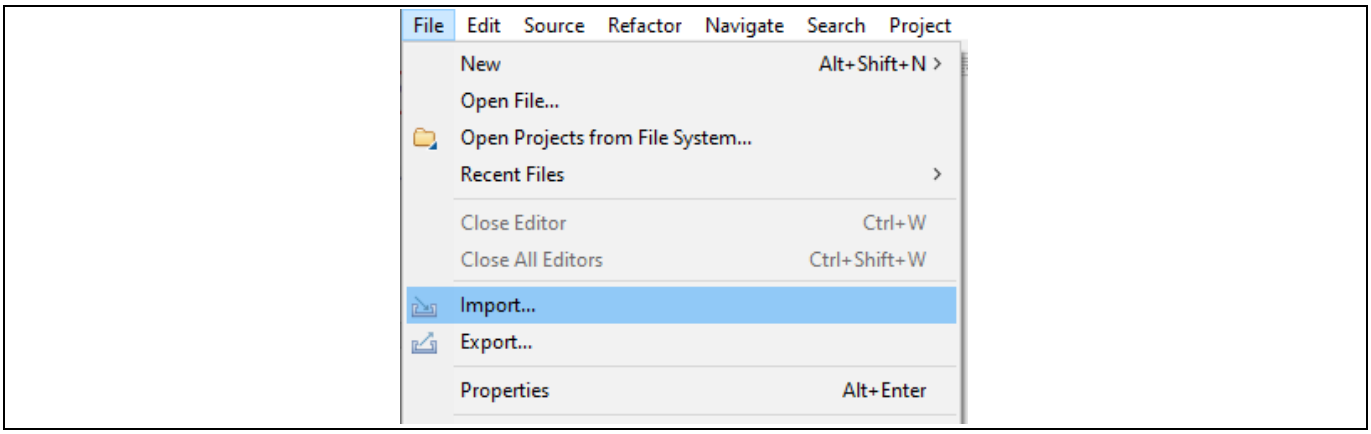


Figure 3-18 Import the Sample Project

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

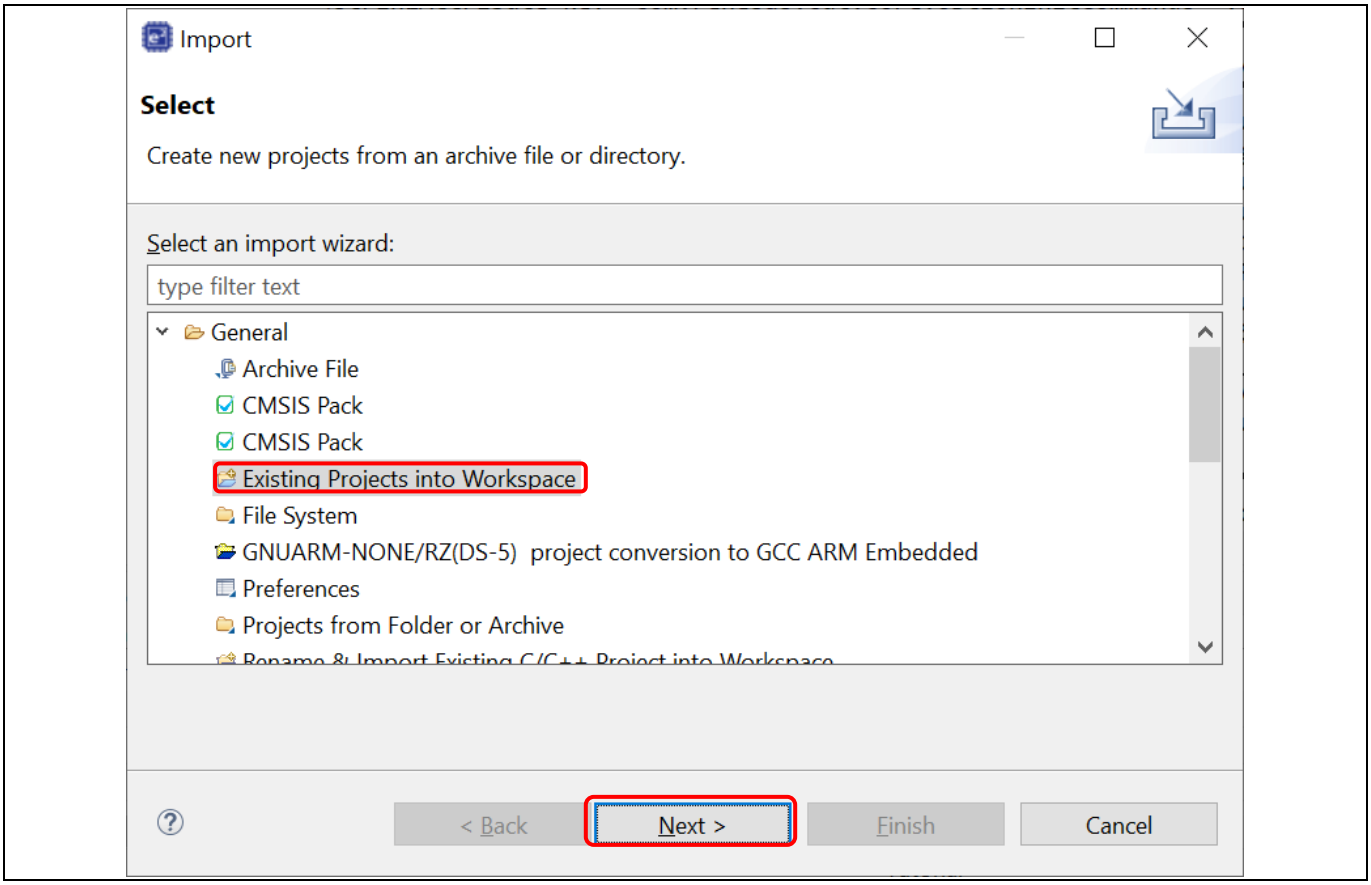


Figure 3-19 Select Import Wizard

5. In the [Import Projects] dialog, select “Select archive file”. Click [Browse] then select the downloaded zip file.

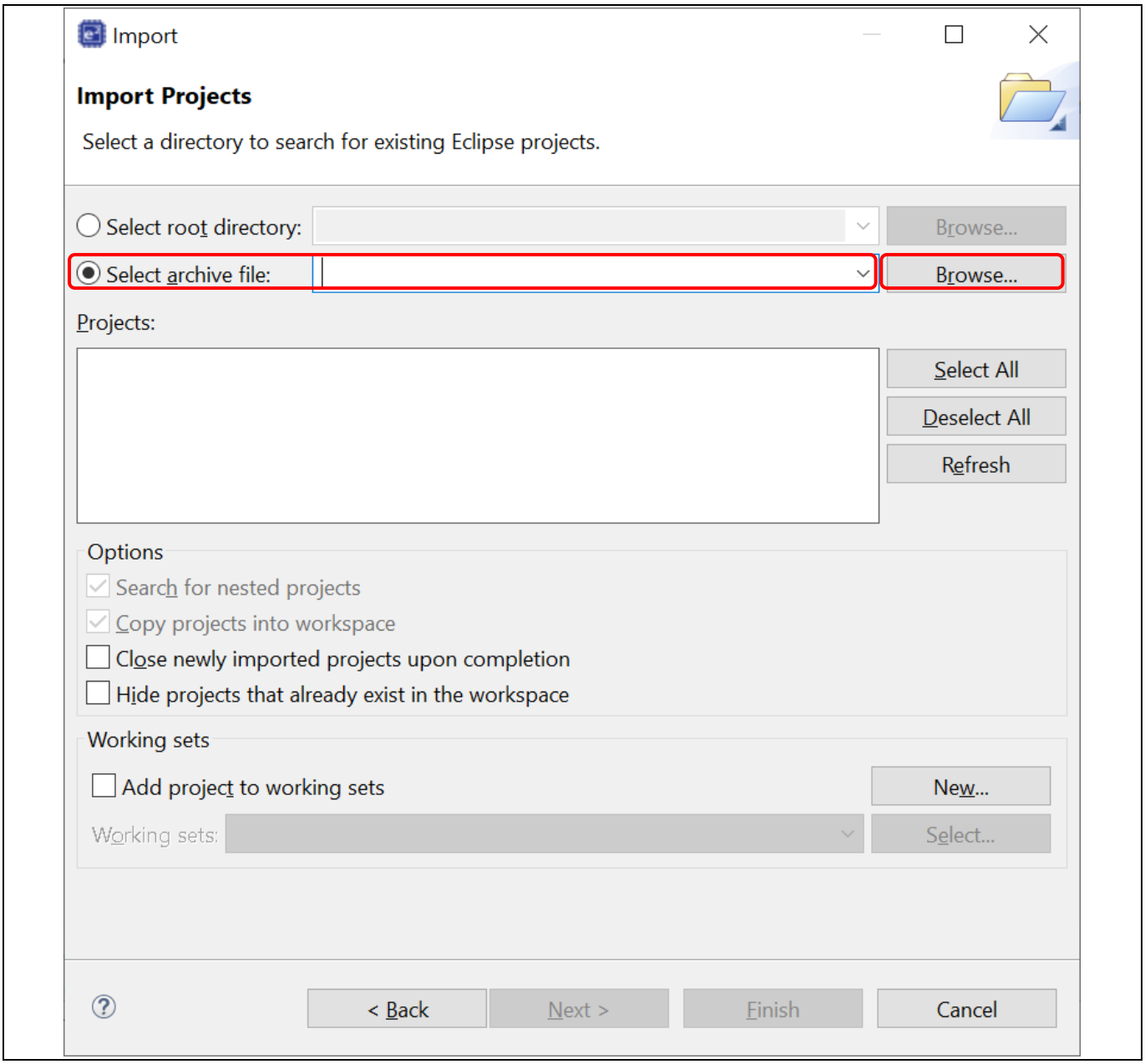


Figure 3-20 Select Project Location to Import

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

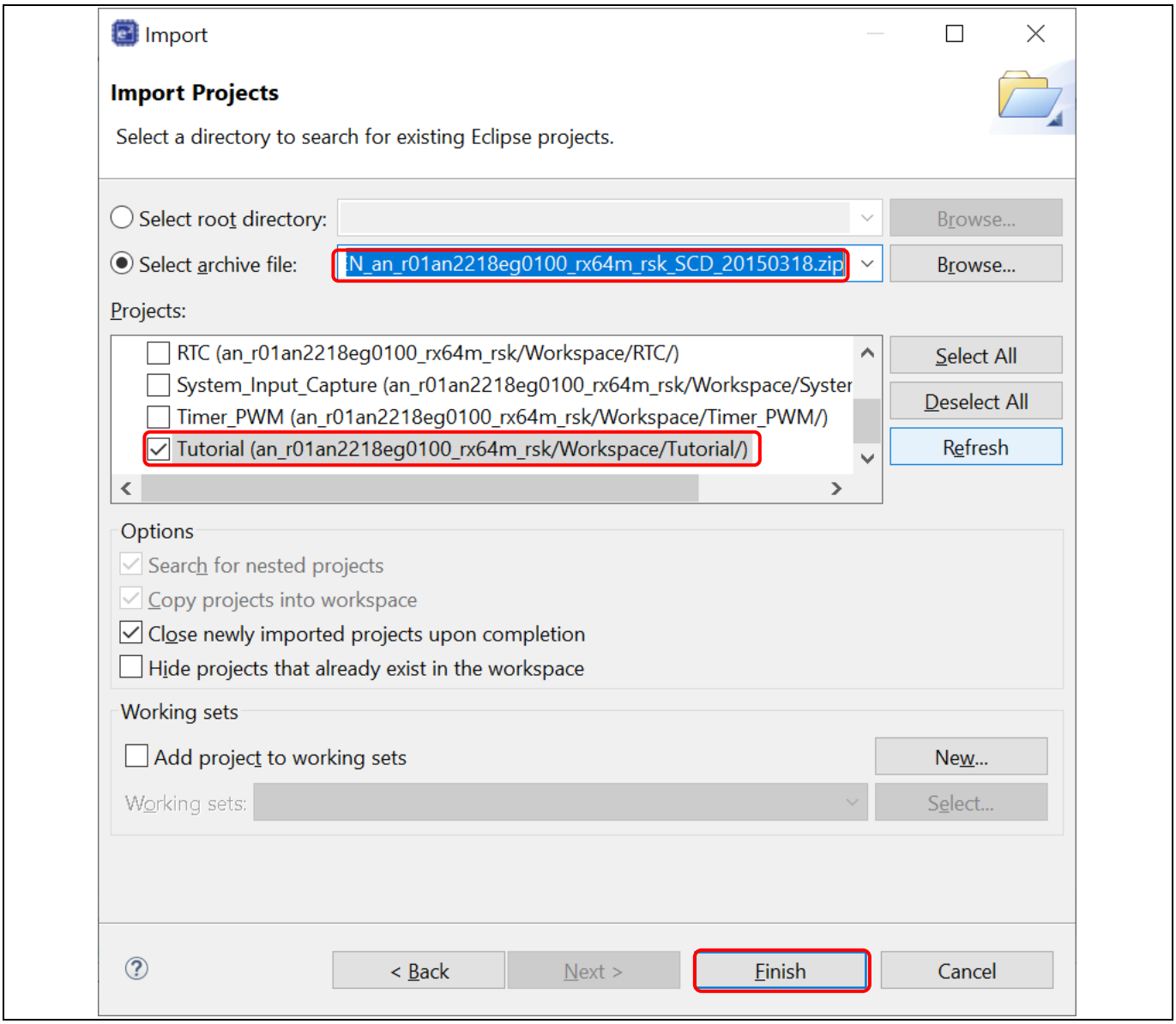


Figure 3-21 Complete Project Import

7. Right-click on the imported project and select “Upgrade Legacy e2 studio Projects…”. If this menu item is not displayed, go to step 9.

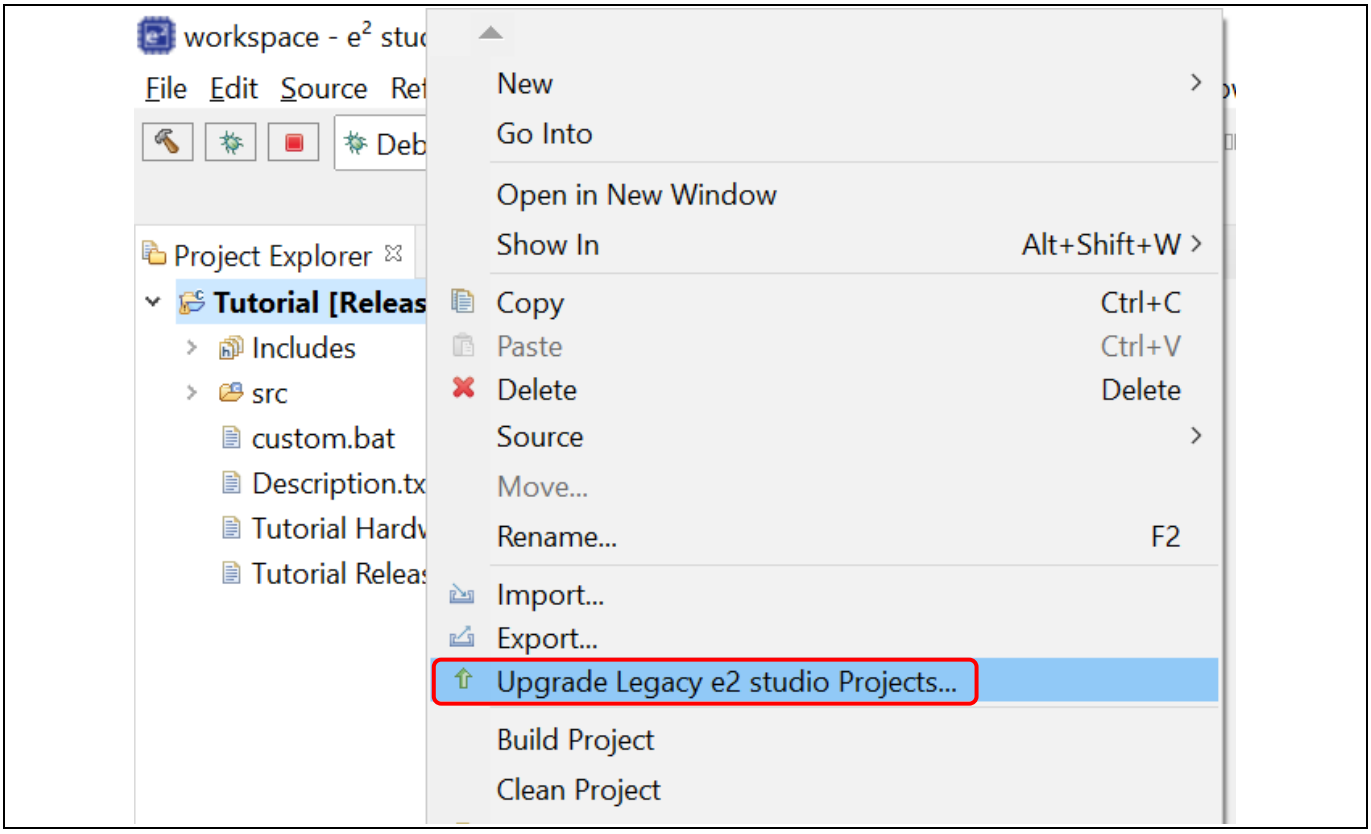


Figure 3-22 Upgrade the Imported Project

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

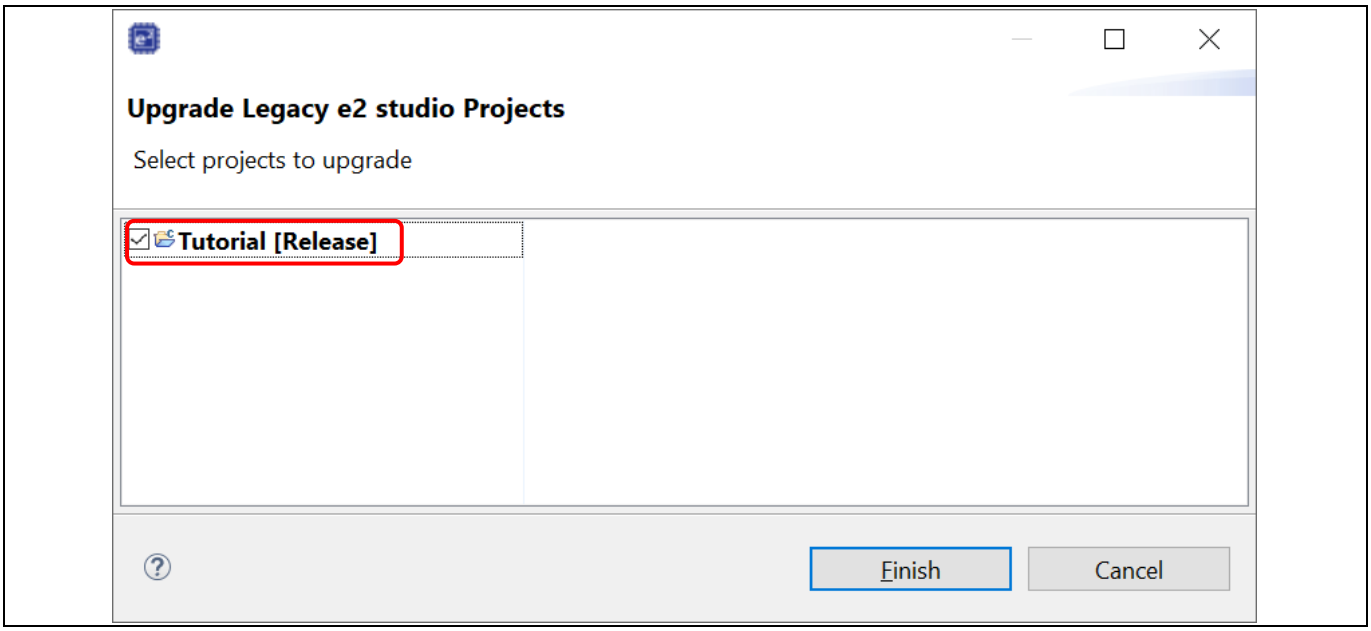


Figure 3-23 Finish Upgrading

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].

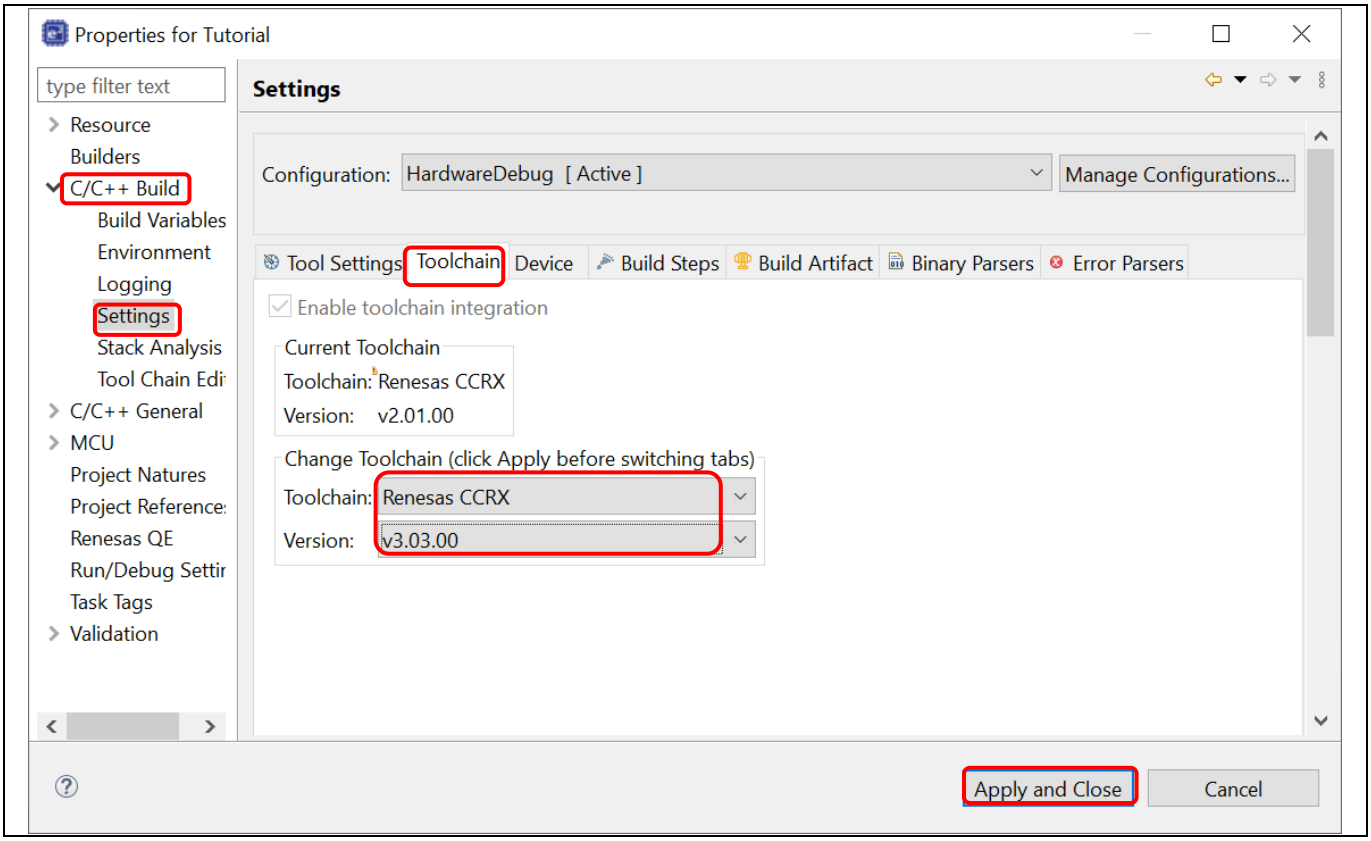


Figure 3-24 Update Project Toolchain

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

3.4.2 Download and Import Sample Projects in the Smart Browser View

You can also download sample projects from the Renesas website through the Smart Browser view. Open the “Renesas Views” -> “Solution Toolkit” -> “Smart Browser” menu (or via “Window” -> “Show View”) to open the view. Right-click on an item listed in the [Application Notes] tab and select “Sample Code (import projects)” to import a sample project into the current workspace. The Application Notes marked as “available” in the “Sample Code” column are provided with sample projects.

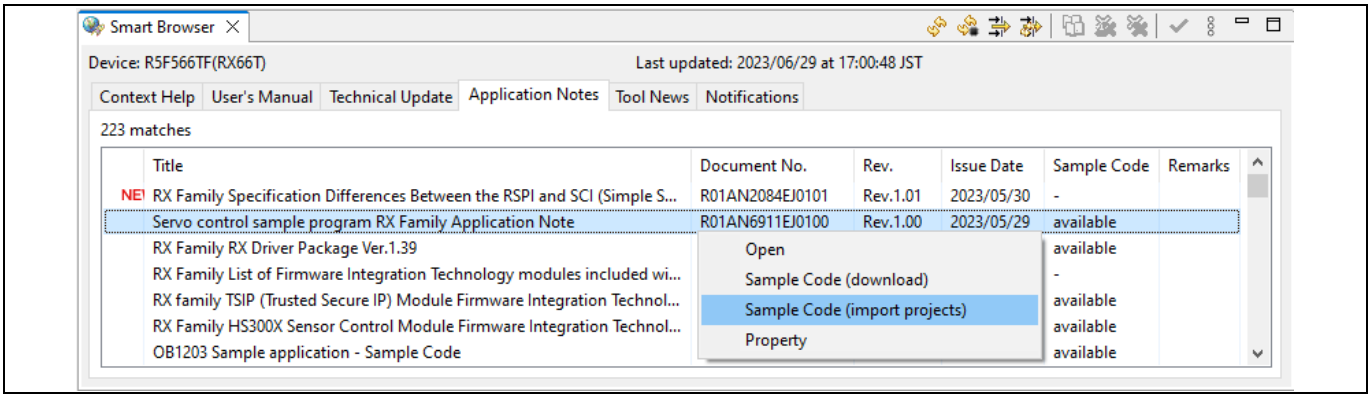


Figure 3-25 Application Notes in the Smart Browser View

3.5 Importing SDK Projects for DA Devices

For details on importing SDK projects or the Web site for downloading toolchains, refer to the following FAQ.

FAQ 3000751 Importing DA SDK project into e² studio

The downloaded SDK can be imported into the current workspace.

The following example describes the methods for importing the software development kit (SDK) that has been downloaded from the product page for the DA14531 or DA14535 device.

1. Download the SDK package from the product page for the DA14531 or DA14535 device.
   Product page for the DA14535: https://www.renesas.com/en/products/wireless-connectivity/bluetooth-low-energy/da14535-smartbond-tiny-da14535-bluetooth-low-energy-53-soc

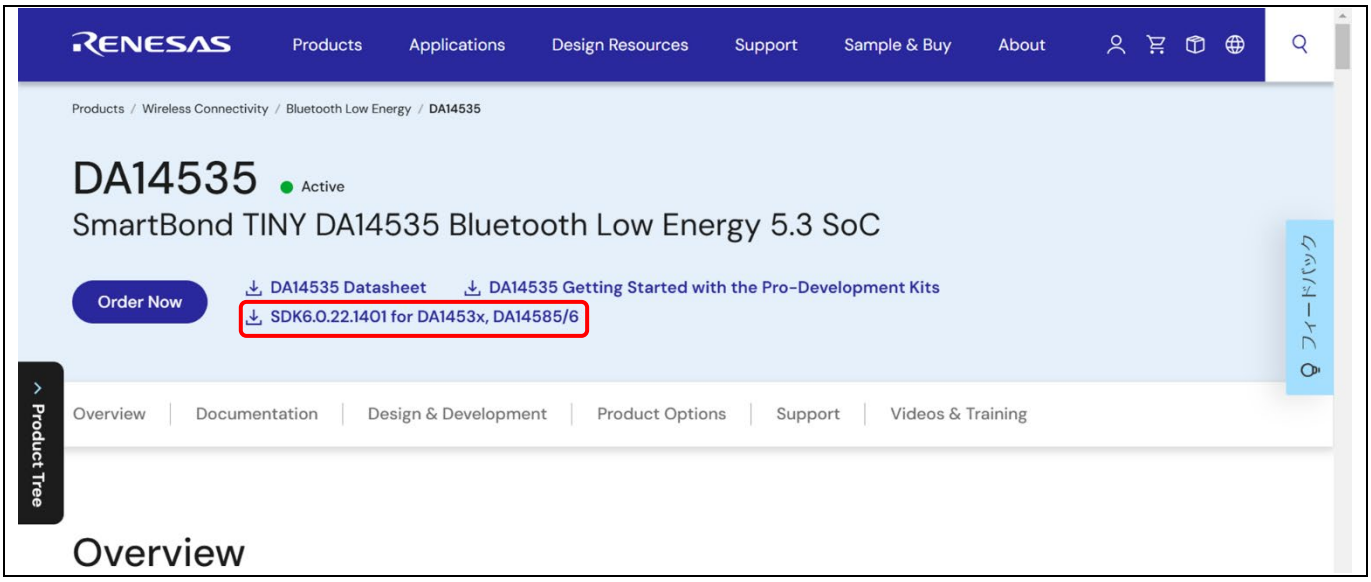


Figure 3-26 Downloading the SDK Package

2. The following shows an example of a downloaded compressed file. Unzip the downloaded SDK package.

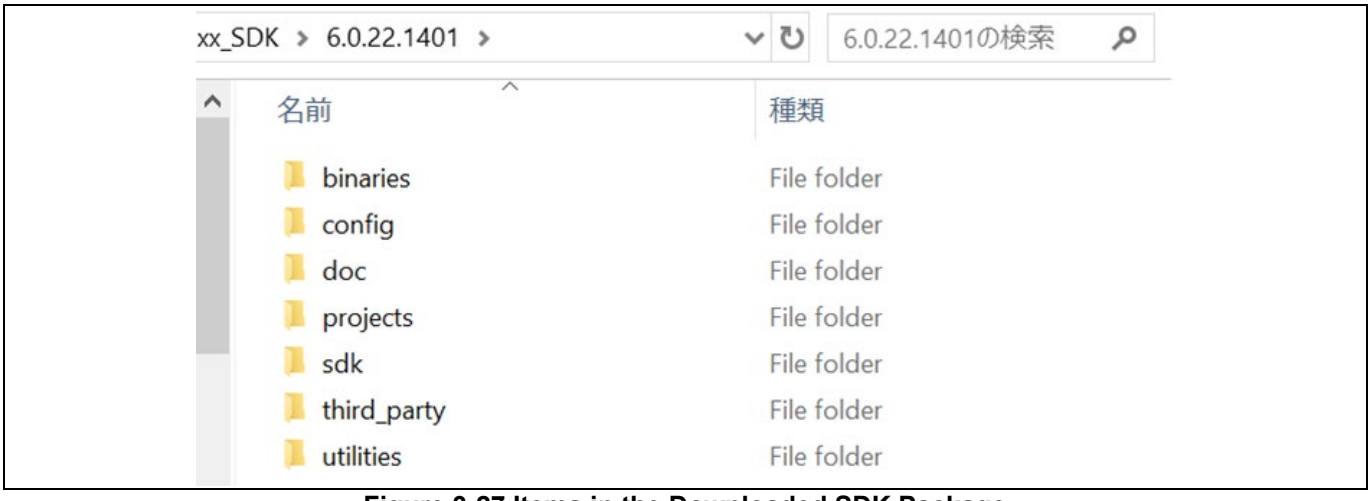


Figure 3-27 Items in the Downloaded SDK Package

3. Start e² studio and open an existing workspace.

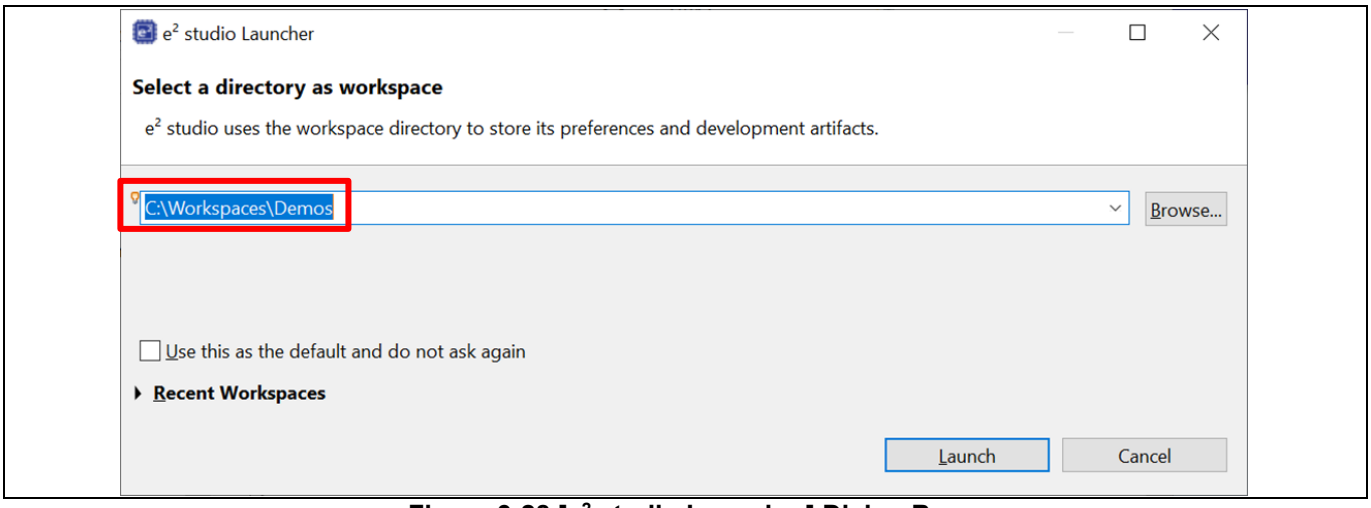


Figure 3-28 [e² studio Launcher] Dialog Box

4. Click on [Import] in the [File] menu.

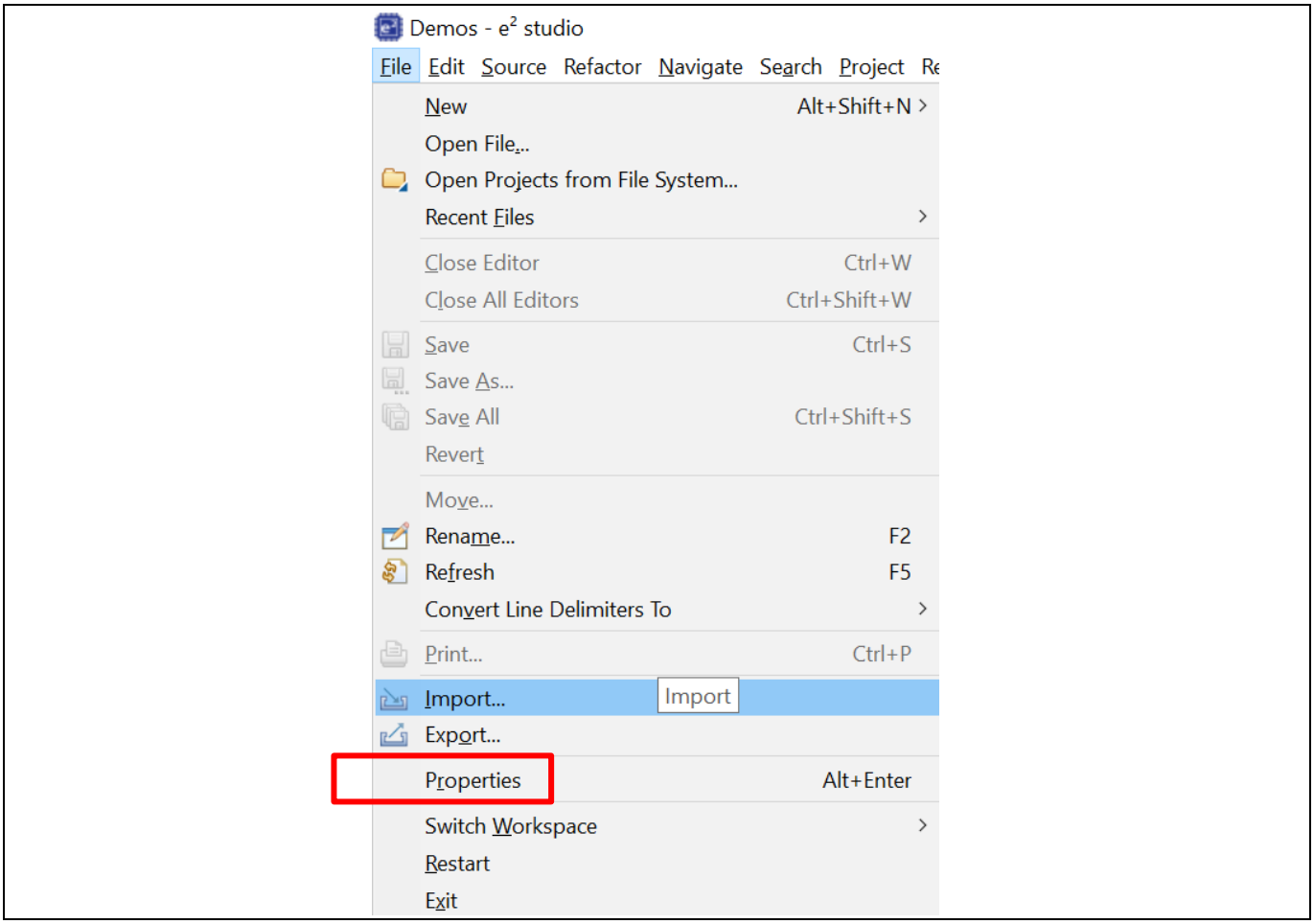


Figure 3-29 Drop-down Menu under [File]

Note: Before importing an SDK project, the relevant toolchain must already have been installed and registered. If you do not know which toolchain the project requires, proceed with the subsequent steps up to Figure 3-35, Specifying the Toolchain, Its Version, and the Target Device, and check the required toolchain and version. For details on installing the toolchain, refer to section 2.4, Installation of Compiler Packages.

5. Select [General] -> [Dialog SDK Project] in the [Import] dialog box and click on [Next].

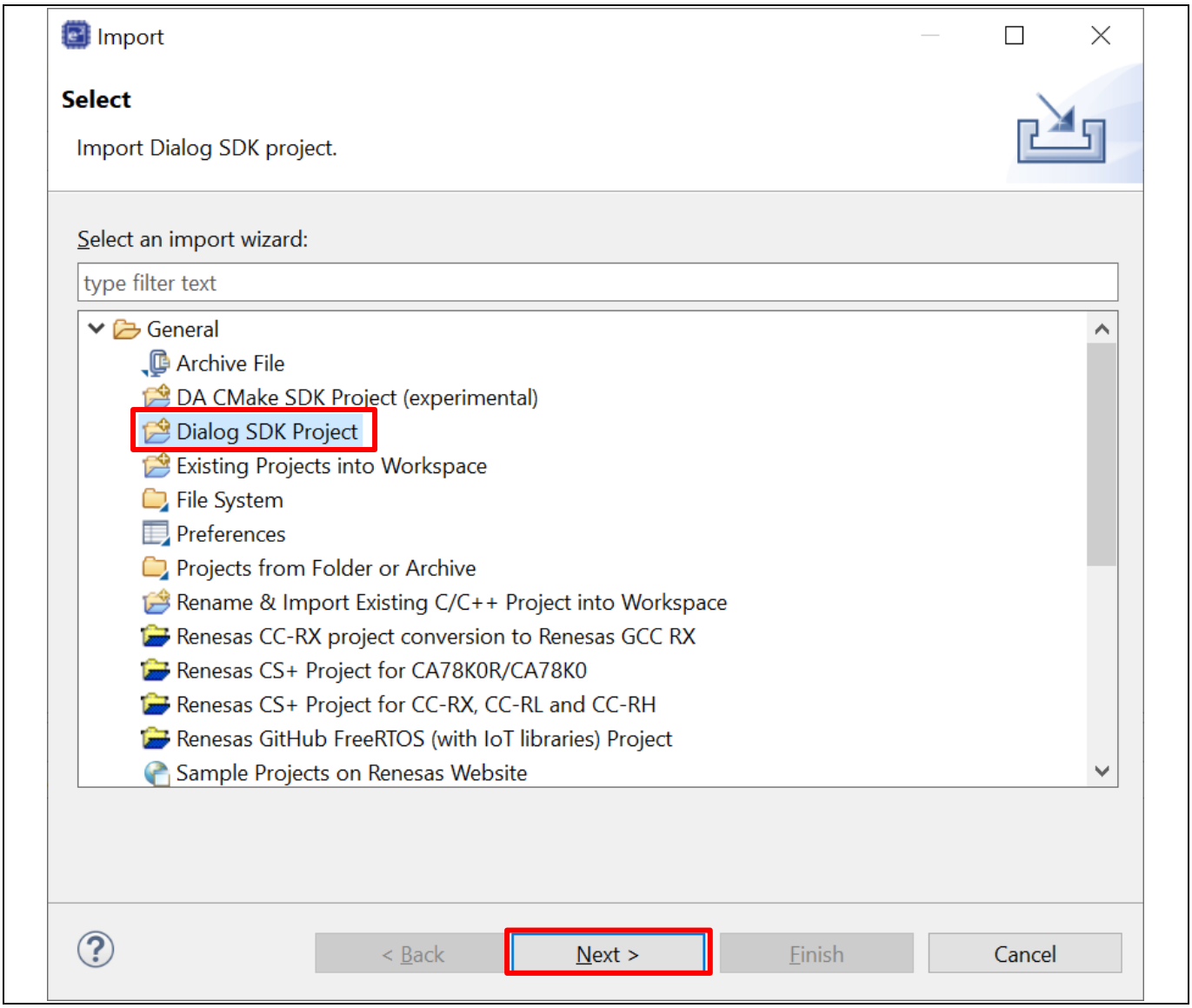


Figure 3-30 Importing [Dialog SDK Project]

Note: When an SDK project is to be installed, do not select [Existing Projects into Workspace]. Instead, use the import options of [Dialog SDK Project] as the [Dialog SDK Project] option has been selected in the [Import] dialog box. In response to doing so, the scripts and other settings required for debugging will be properly configured; e.g. the launcher for configuring debugging will correctly be set.

6. Select the folder where the SDK was unzipped as [Select SDK root directory:].
   Use [Browse] to select the SDK root folder.

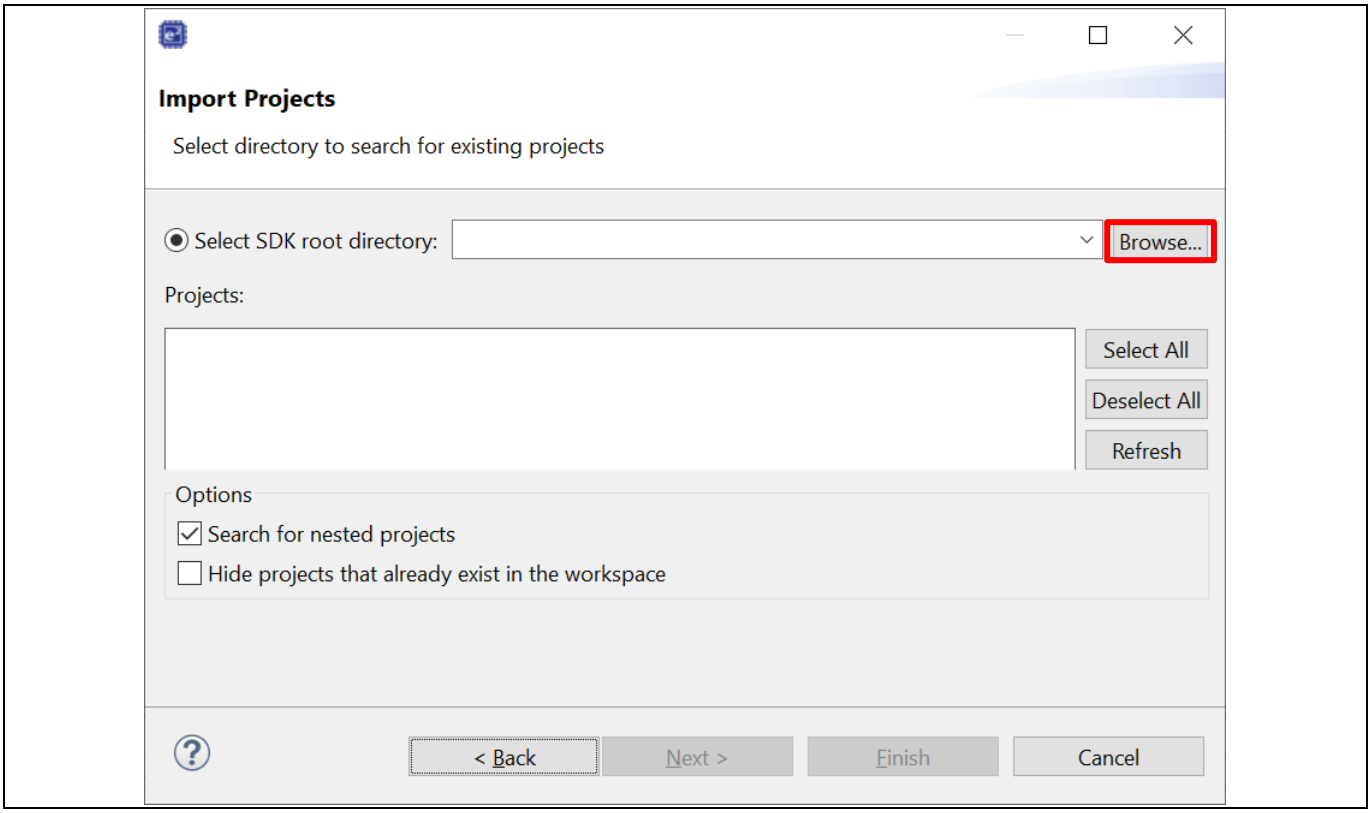


Figure 3-31 Selecting the SDK Root Folder

7. Select the SDK root folder so that the [config] and [projects] subfolders are included.

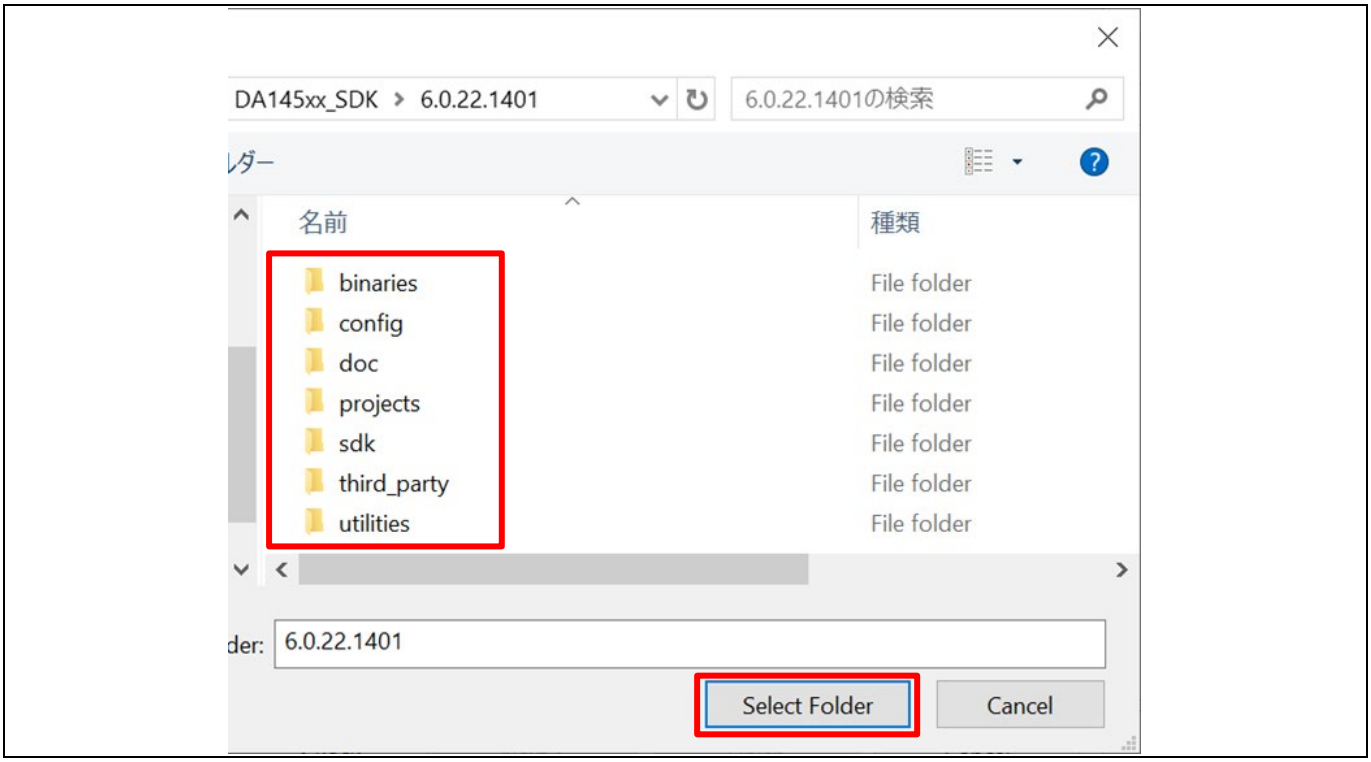


Figure 3-32 Selecting an SDK Root Folder Including the Required Subfolders

8. A list of sample projects is shown in the [Project:] panel.
   Select the project to be imported. Here, select the [prox_reporter] project for [e2 studio].
   Click on [Next] to go to the next screen.

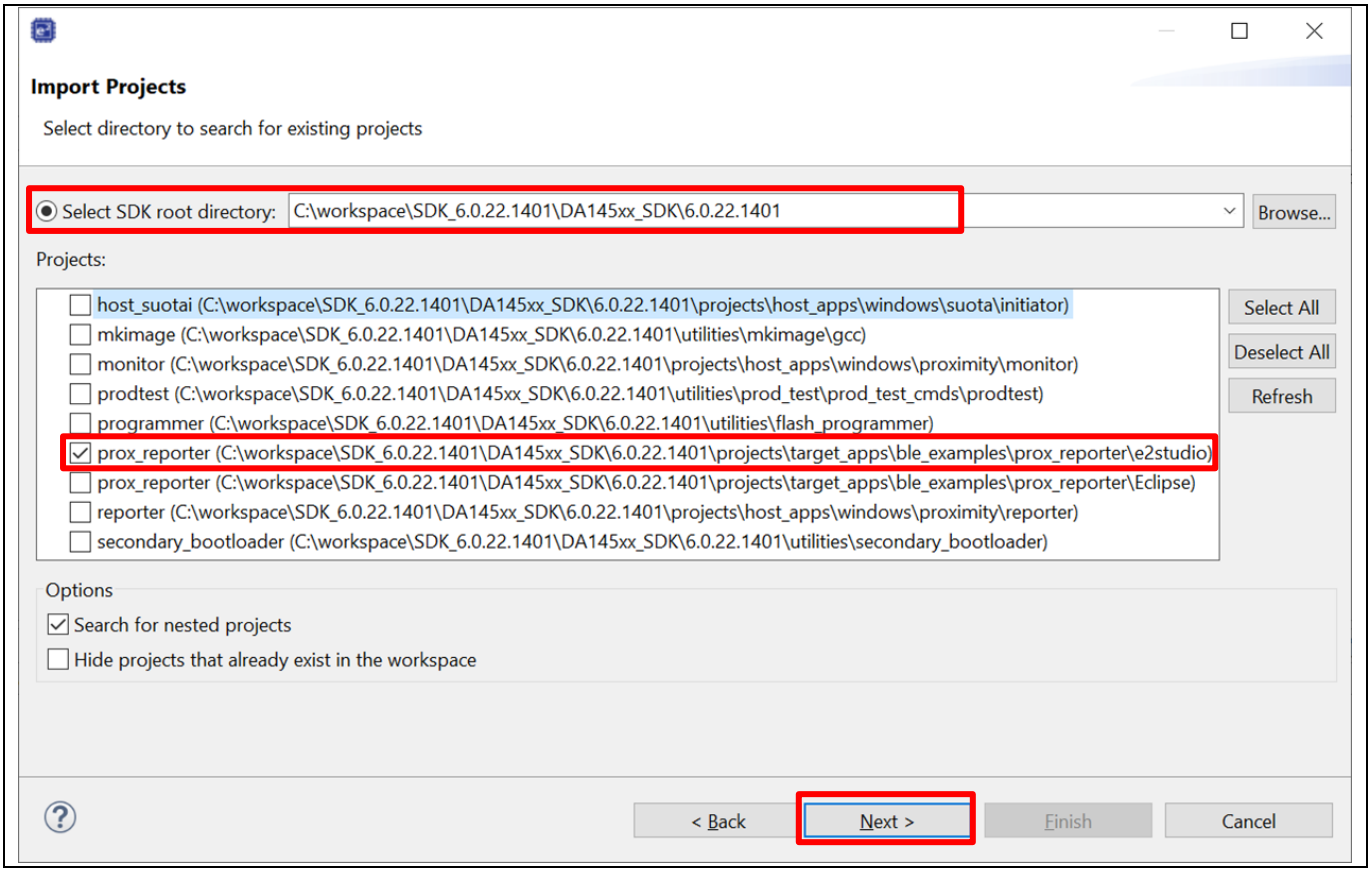


Figure 3-33 Selecting the SDK Project

Note: If the selected SDK root folder is incorrect, a warning message will appear. Re-select the SDK root folder that includes the required setting files and subfolders for the project from [Browse].

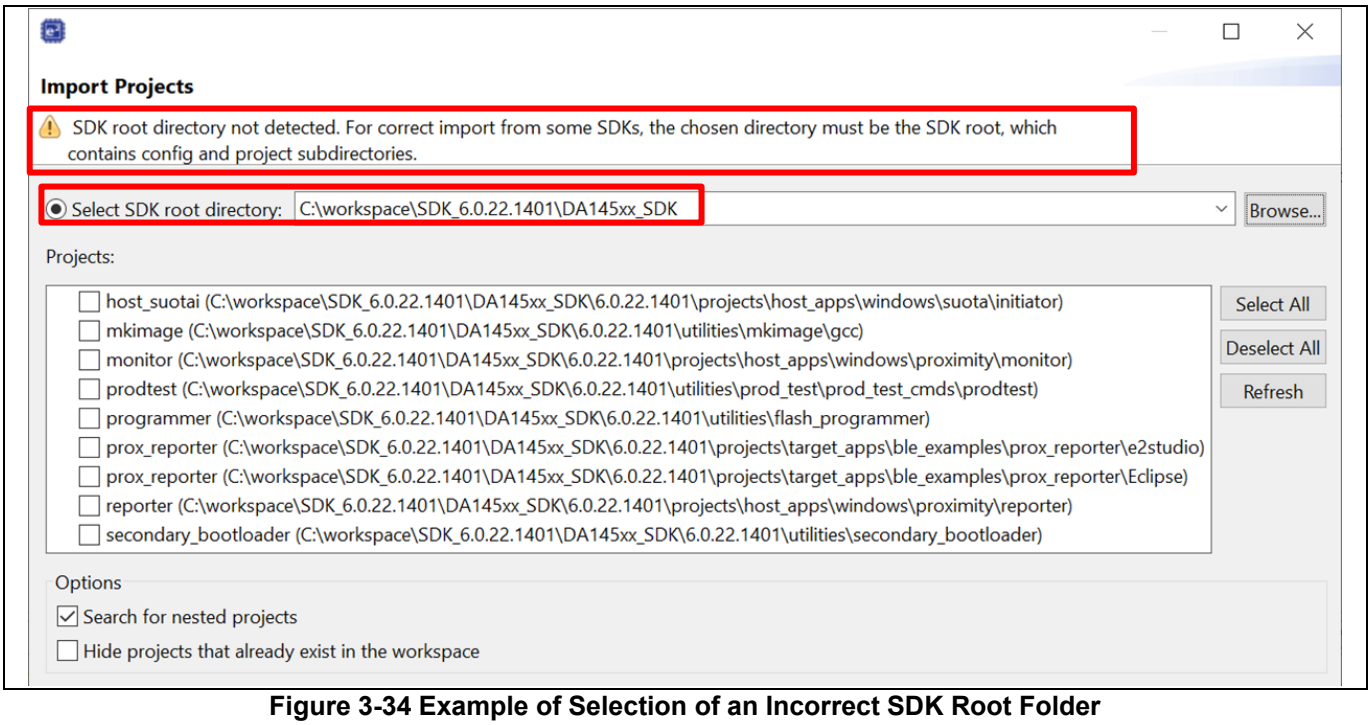


Figure 3-34 Example of Selection of an Incorrect SDK Root Folder

9. Specify the toolchain, its version, and the target device.
   The target device can be selected by clicking on […].

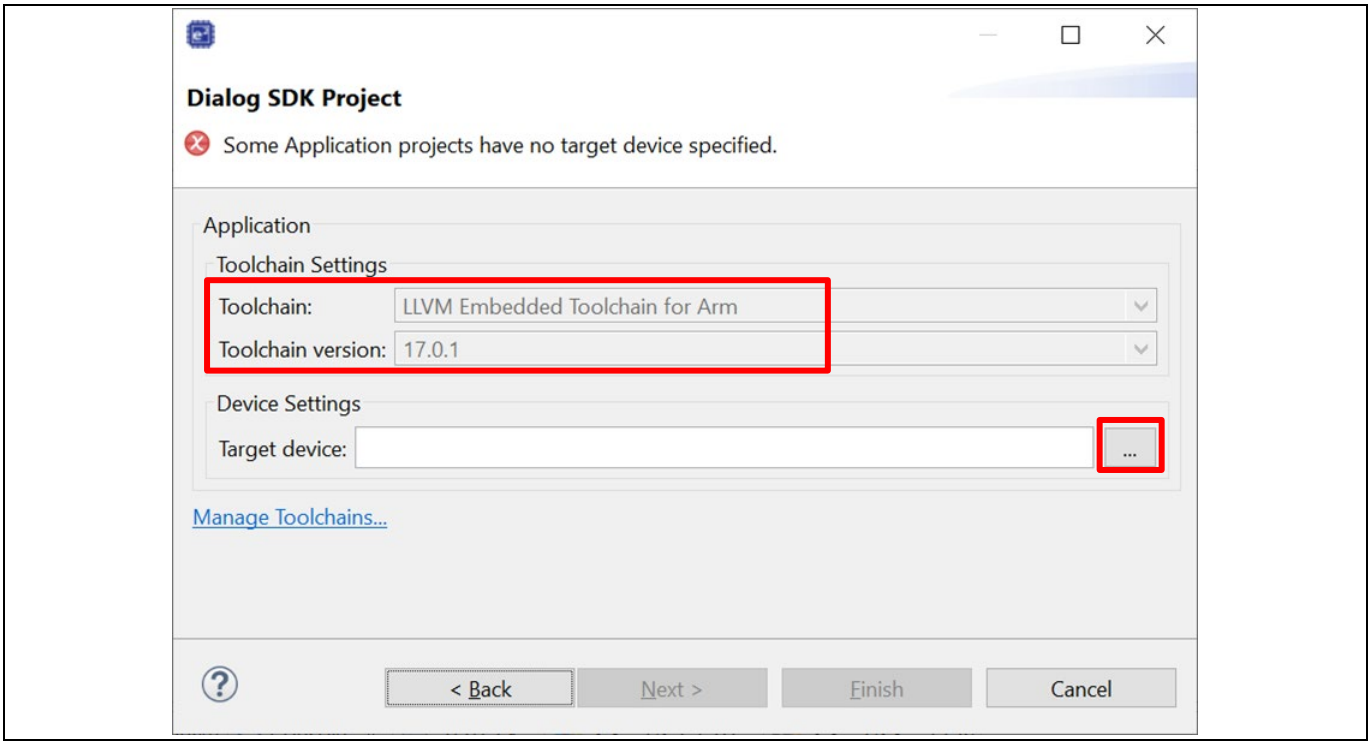


Figure 3-35 Selecting the Toolchain, Its Version, and the Target Device

10. Select the target device.

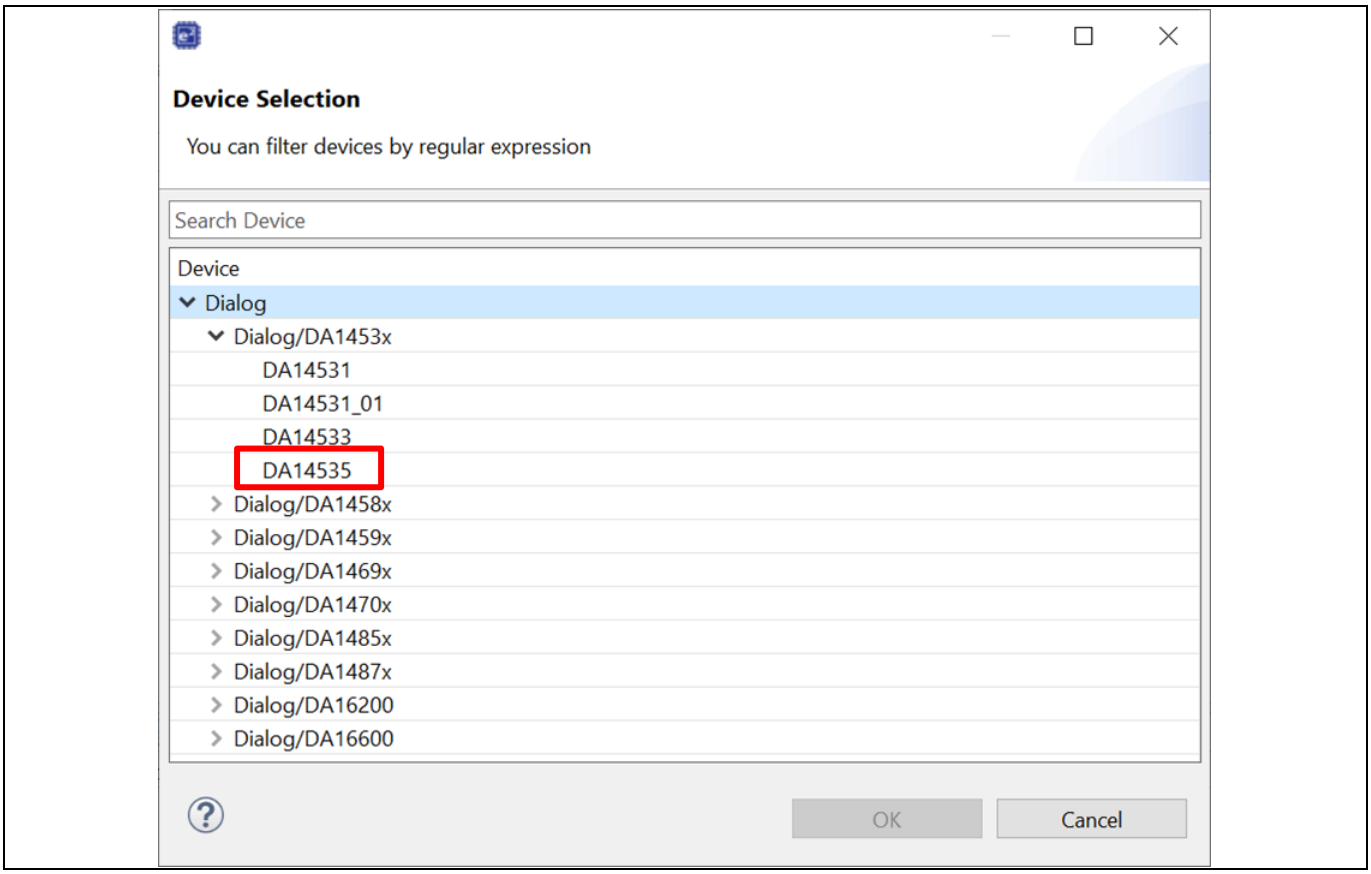


Figure 3-36 Selecting the Device

11. After you have selected the target device, click on [Finish] to import a project.

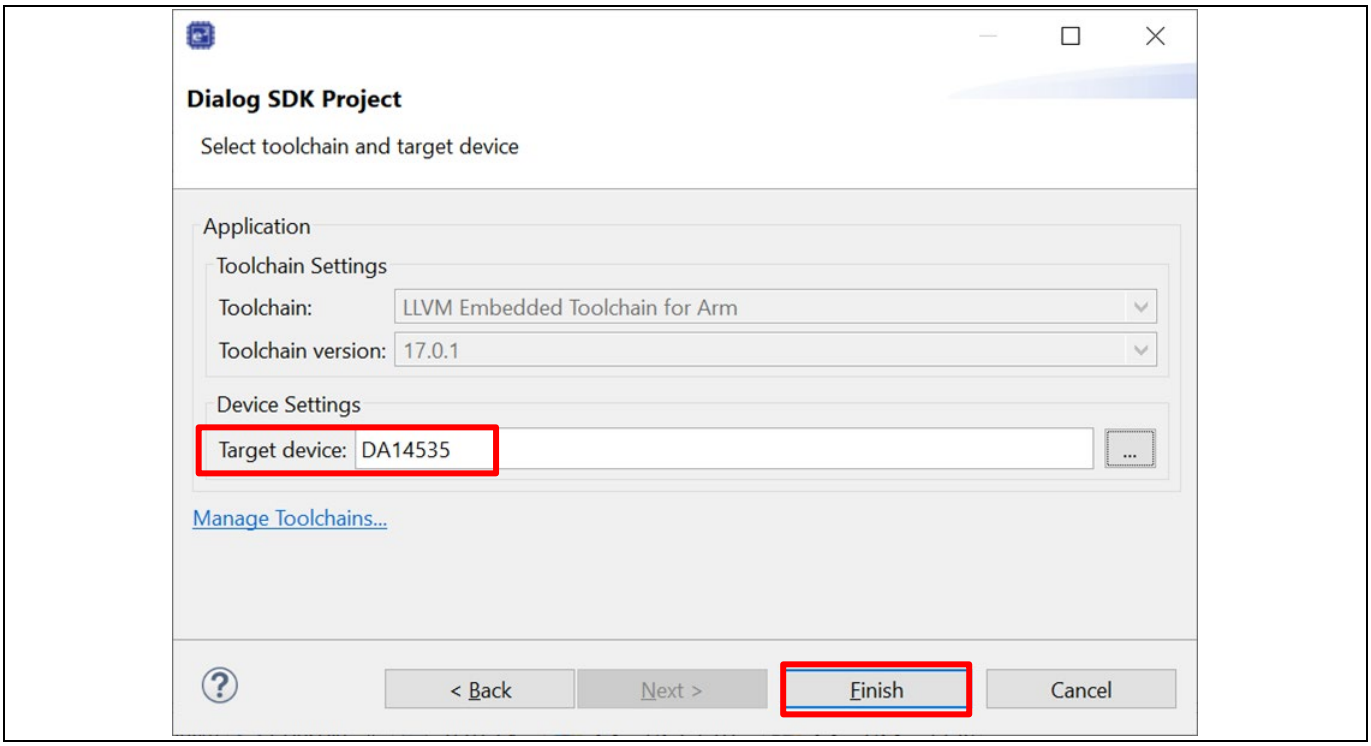


Figure 3-37 Completion of Importing the SDK Project

12. Check that building a project is successful.
    Run building by clicking on the icon (downward arrow) -> [Debug_RAM_DA14535].
    On completion of building, the message shown below appears.

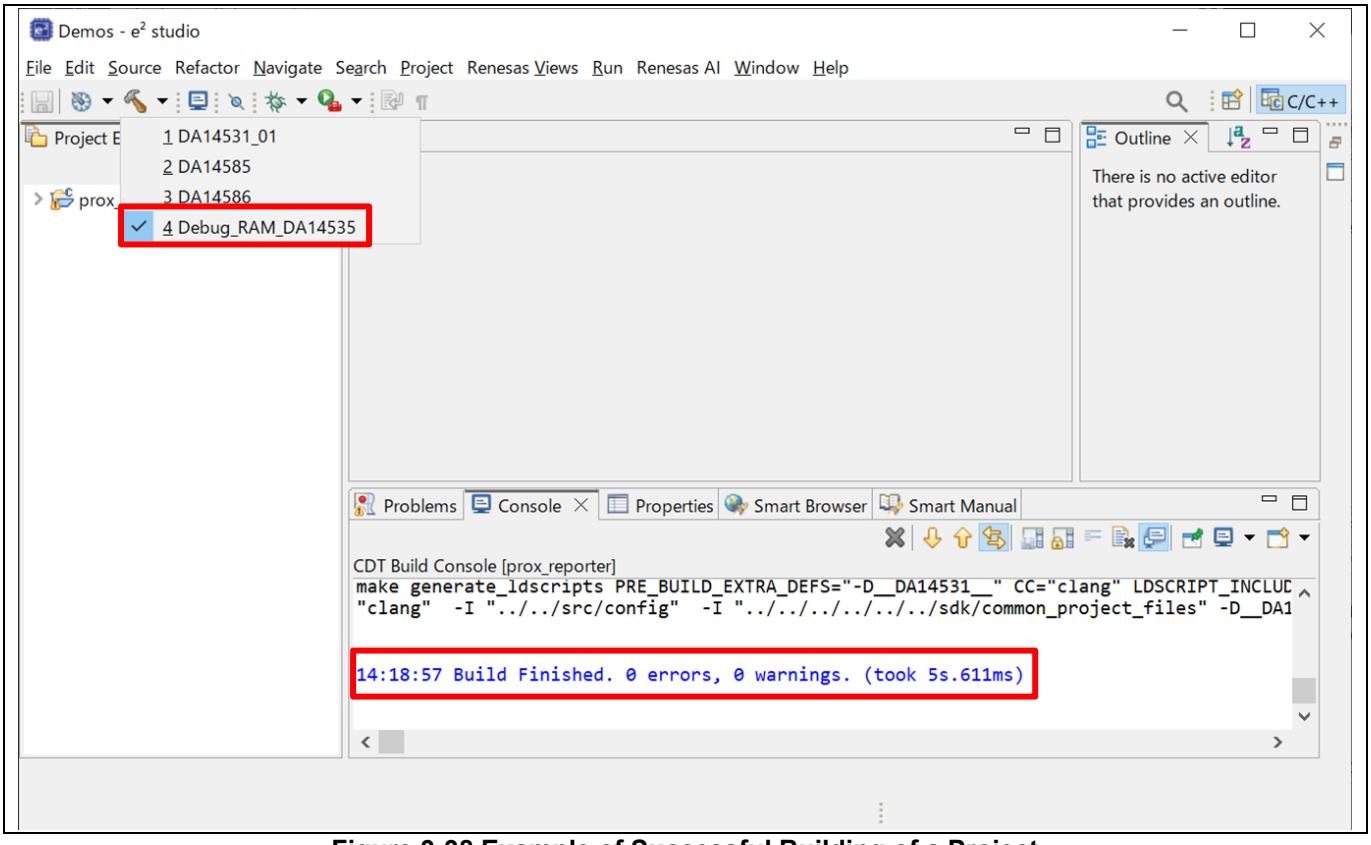


Figure 3-38 Example of Successful Building of a Project

4. Build

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

4.1 Build Option Settings

A new project built with the default option can work properly. However, if users 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] to open the Properties window.

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

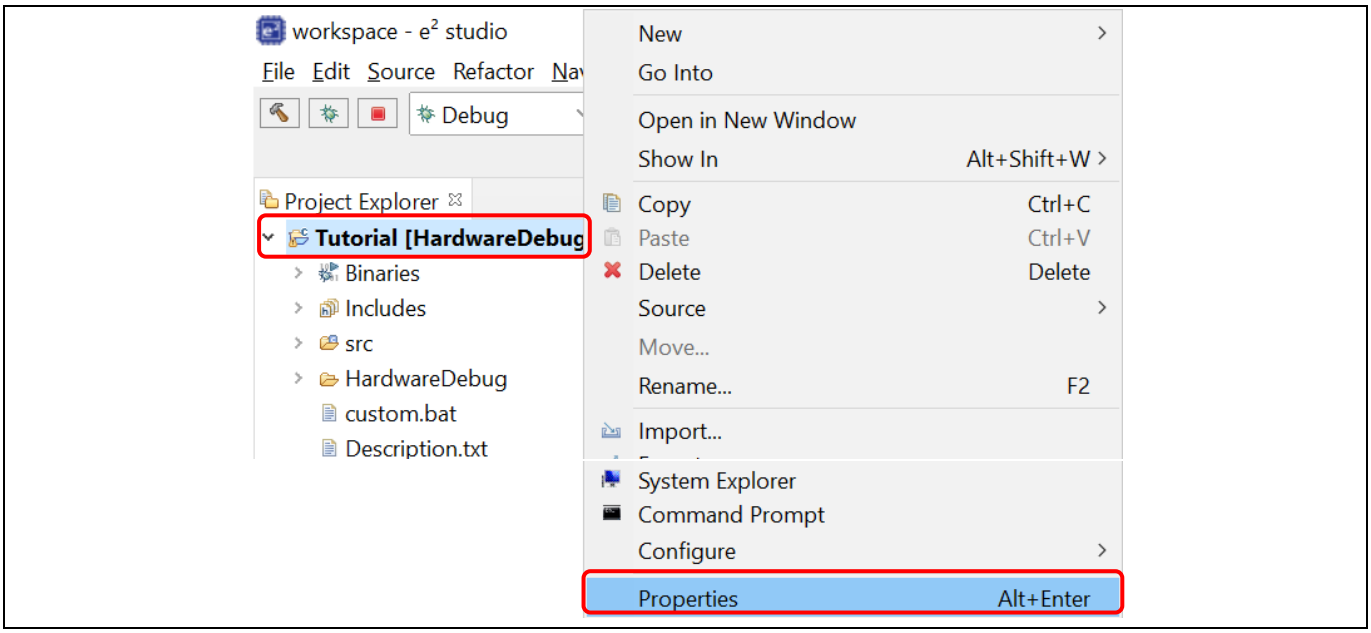


Figure 4-1 Open the Properties Window

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

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

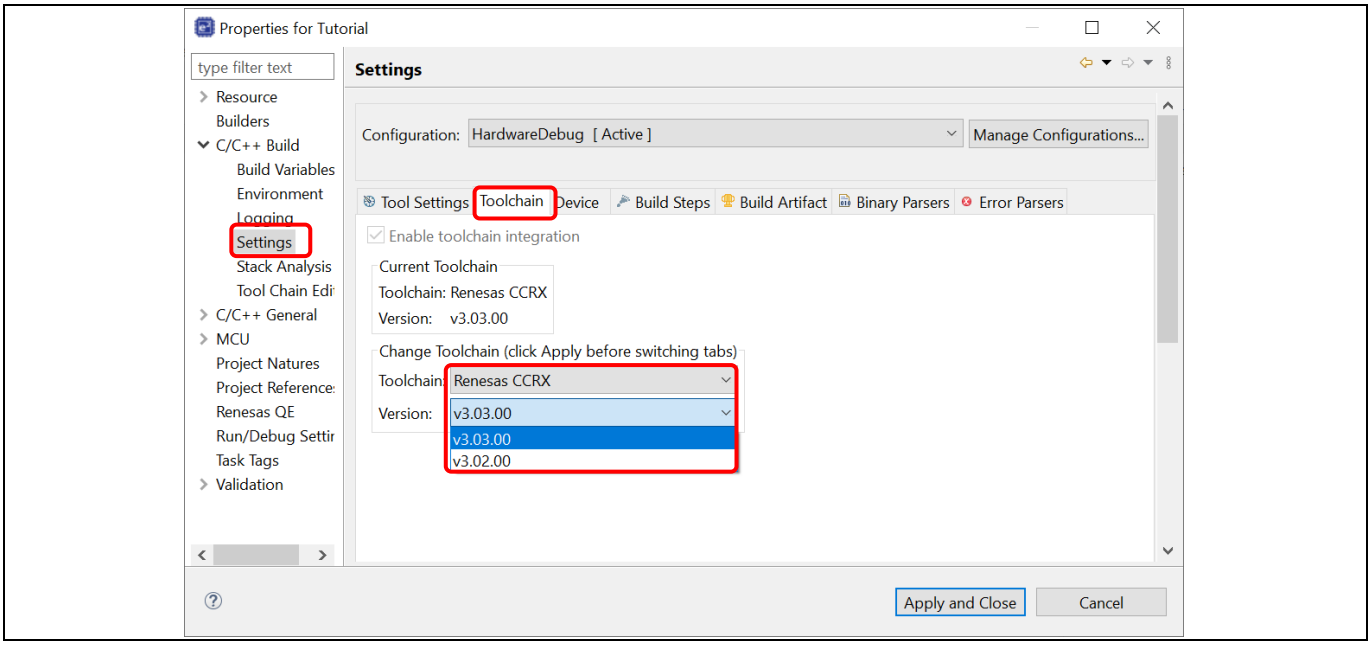


Figure 4-2 Change Toolchain Version

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

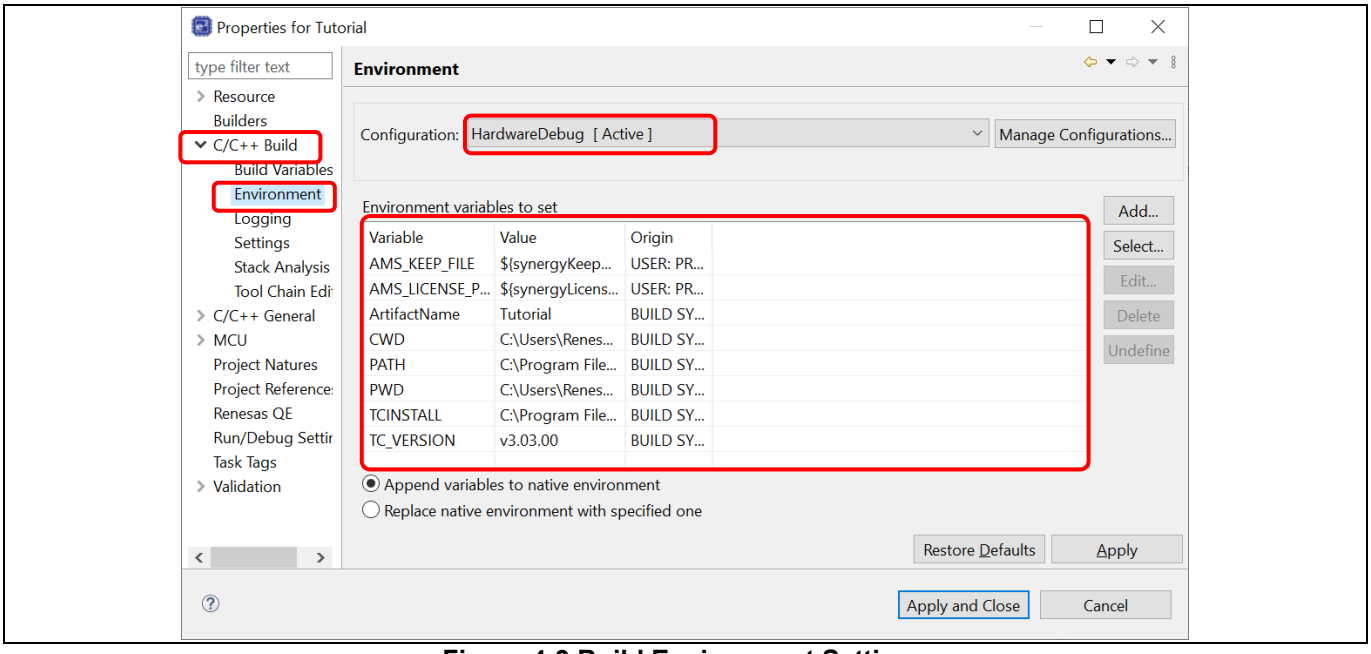


Figure 4-3 Build Environment Settings

4. Setting Build Options

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

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

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

The “Build configuration” can be switched via the “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.

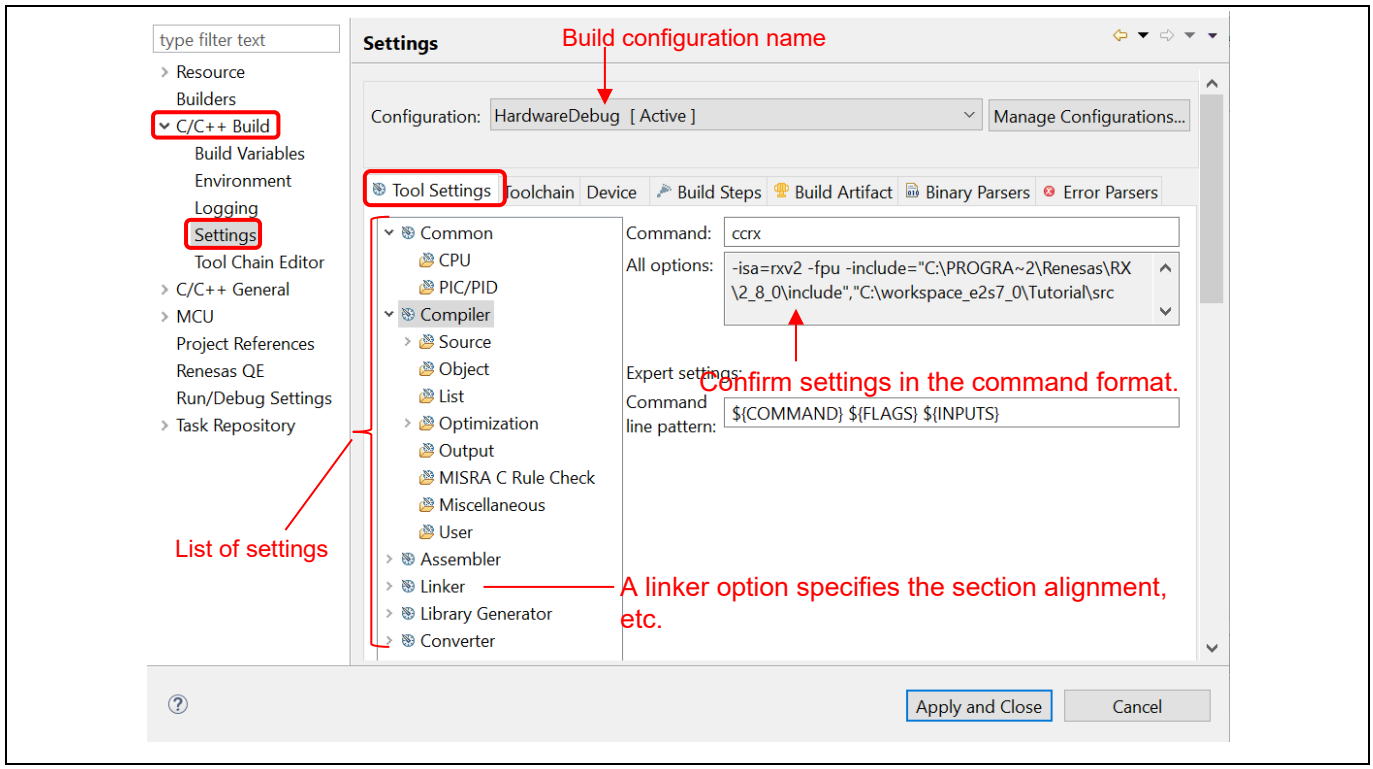


Figure 4-4 Build Option Settings

Details of build options are described in the compiler user’s 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].

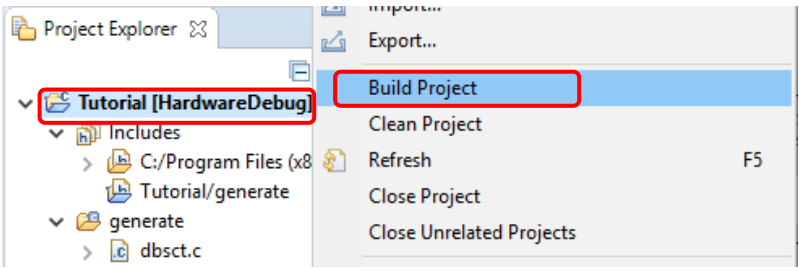


Figure 4-5 Build a Sample "Tutorial" Project

2. Check that the [Console] pane shows the '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 the ELF/DWARF format (*.abs) used for debugging. Because GDB supports a load module format with the different ELF/DWARF specification (*.x or *.elf), hence "Tutorial.abs" has to be converted to "Tutorial.x" for debugging in the e² studio IDE.

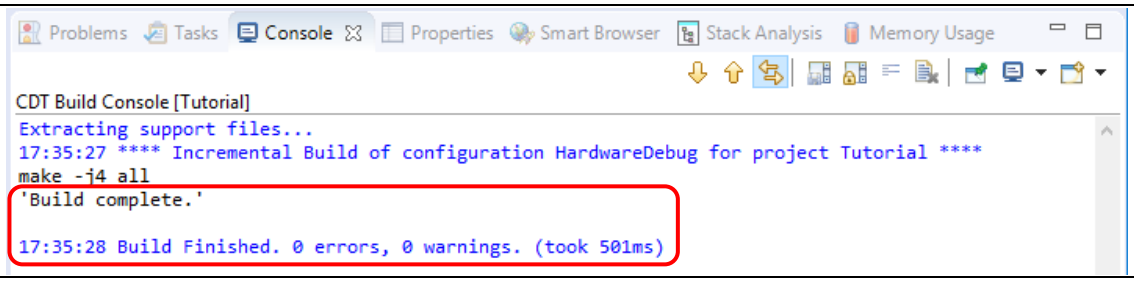


Figure 4-6 Project is Built Successfully.

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.

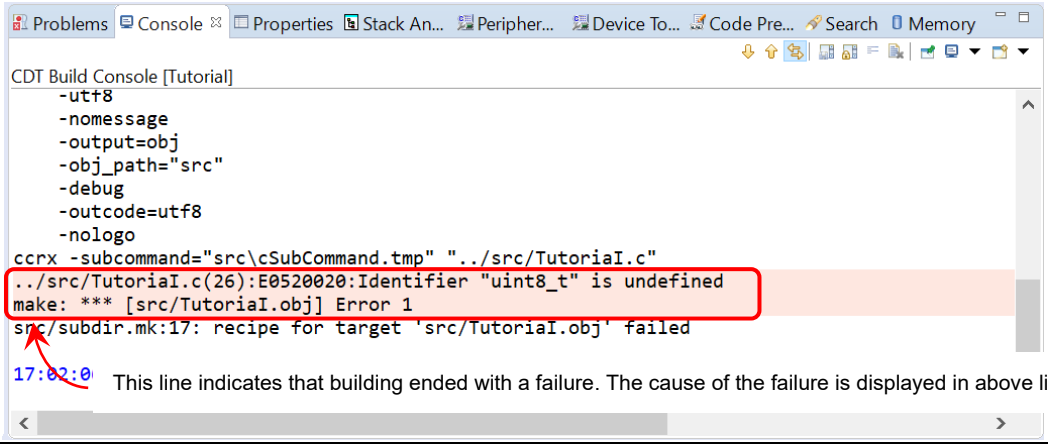


Figure 4-7 Unsuccessful Build Shows Error Messages

4.3 Export Build Configuration Settings

The Project Reporter feature can export project and build configuration settings from the 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 [Renesas C/C++ Project Settings] -> [Save build settings report] to save the build settings report.

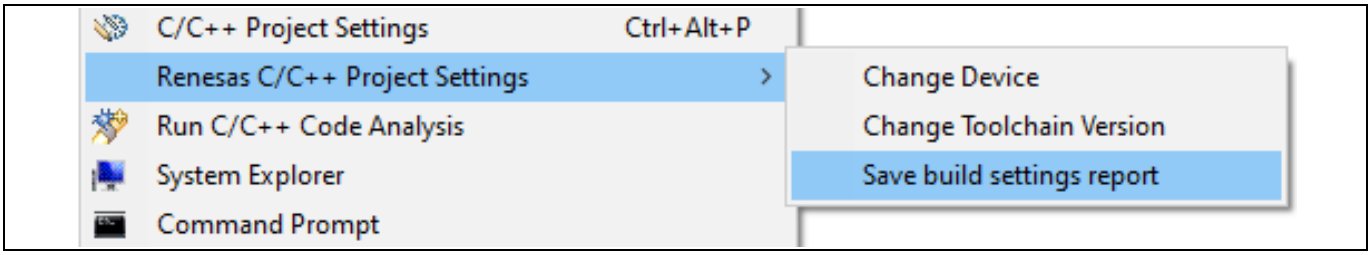


Figure 4-8 Project Reporter

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 the hardware configurations of the E2 emulator Lite and RSK RX64M board.

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

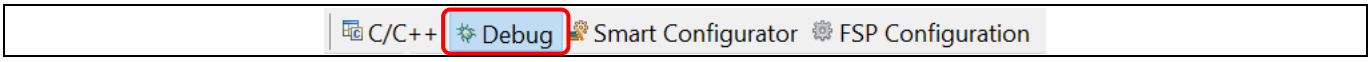


Figure 5-1 Switch To [Debug] Perspective

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 users to perform the specific task.

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

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

Note: For more information on debug, please refer to “e2 studio User Guide” 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 the “Tutorial” project in the [Project Explorer] pane to set focus.

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

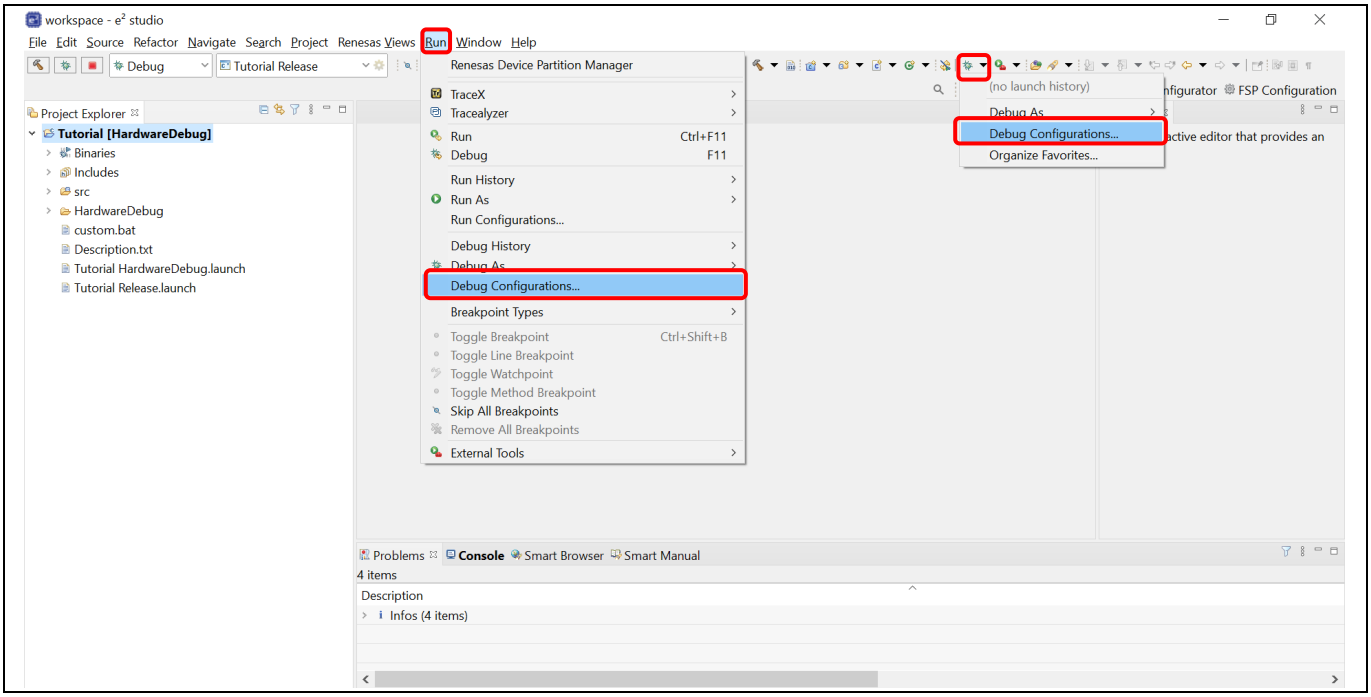


Figure 5-2 Open the Debug Configurations Window

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

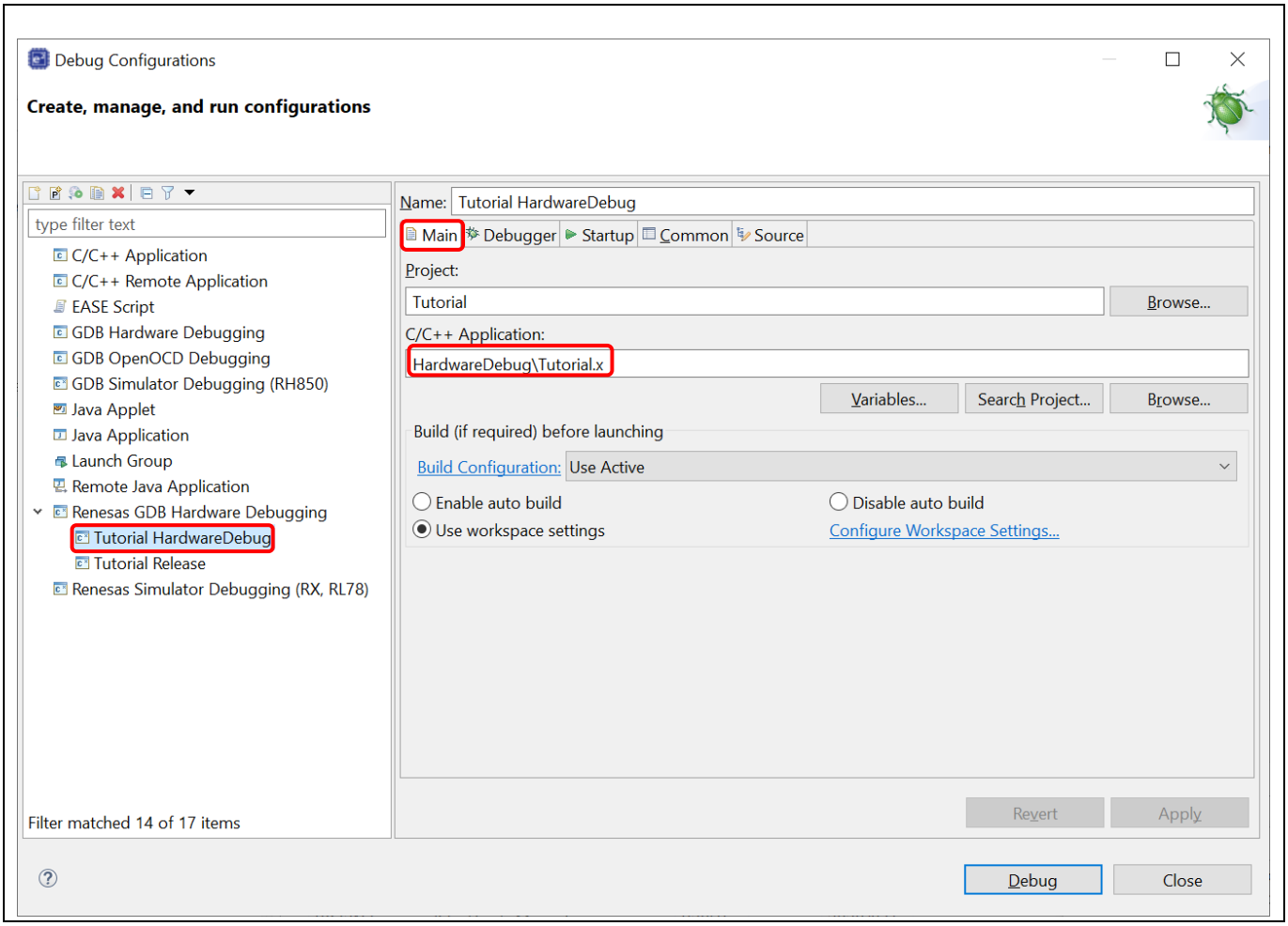


Figure 5-3 Select the Load Module

3. Switch to the [Debugger] tab, set “E2 Lite (RX)” as the debug hardware and “R5F564ML” as the target device.

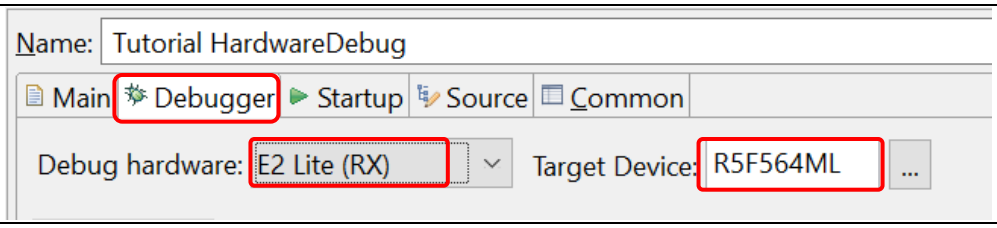


Figure 5-4 Select the Target Device

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 the E2 emulator Lite 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] = “6.00”

— Hot plug = “No”

The hot plugin feature is only available with the device which has the capability. Please refer to the 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 an ID code area, although the 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, pay attention to verify the board and emulator settings before connection.

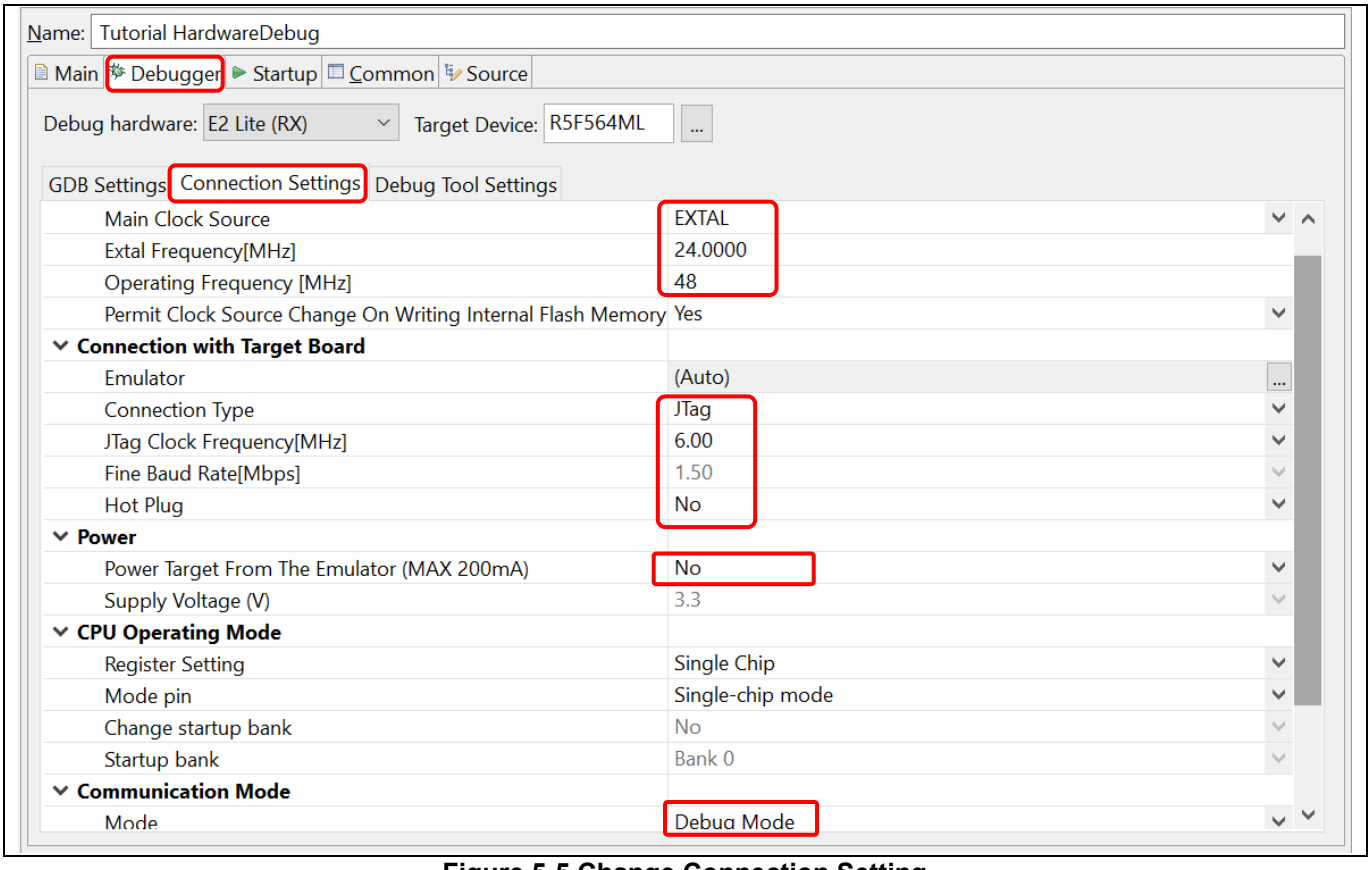


Figure 5-5 Change Connection Setting

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

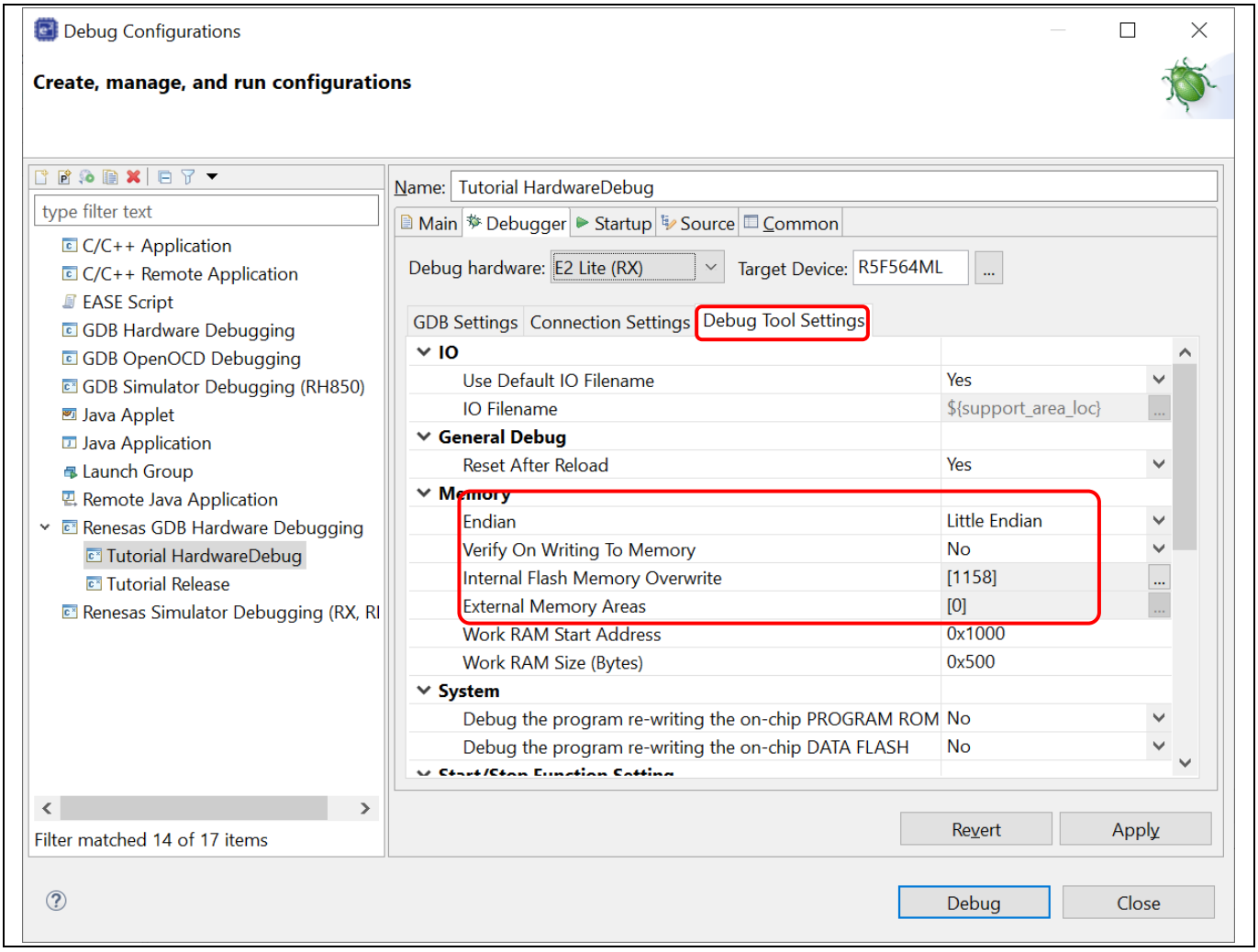


Figure 5-6 Change Debug Tool Settings

- Memory
  - Endian = “Little Endian”

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

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

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

6. Click the [Apply] button to confirm and save the settings. Then click [Debug] to connect the debugger and start downloading the load module.

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

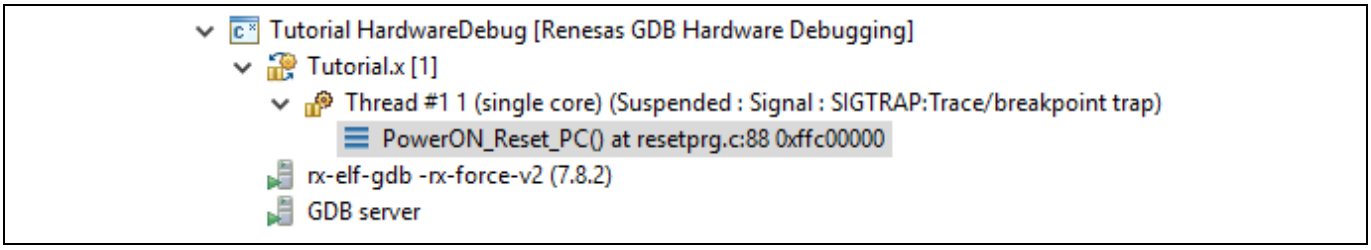


Figure 5-7 User Target Connection in the [Debug] View

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 the “Debug Configurations” window.

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

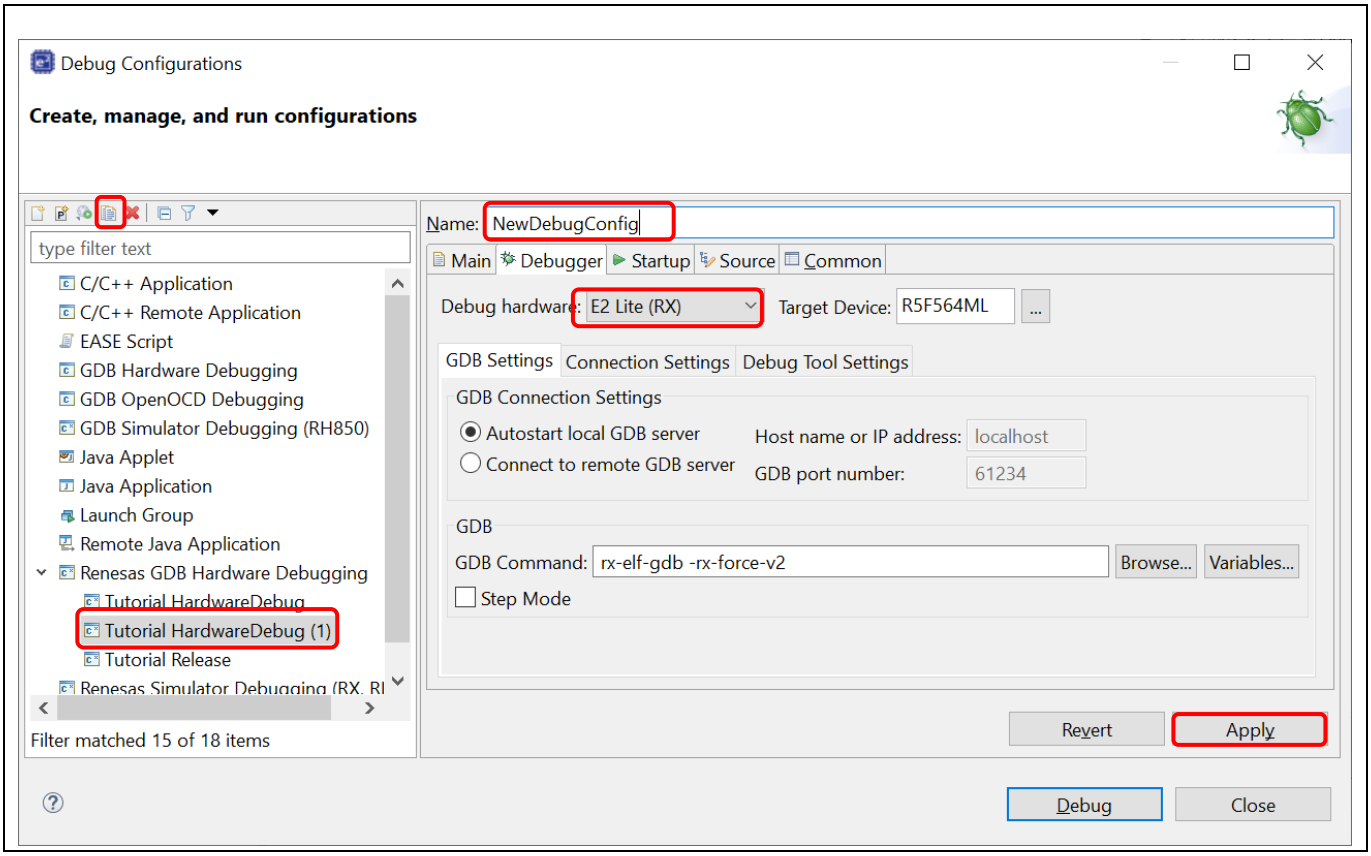


Figure 5-8 Duplicate a Selected Debug Launch Configuration

Note: If no debug configurations have been created and duplication is not possible, then right-click the load module (extension *.x or *.elf) in Project Explorer to start the debugger with "Debug As" → "Renesas GDB Hardware Debugging" (or “Renesas Simulator Debugging” for the simulator), and then back to the Debug Configurations dialog to make the required settings.

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.

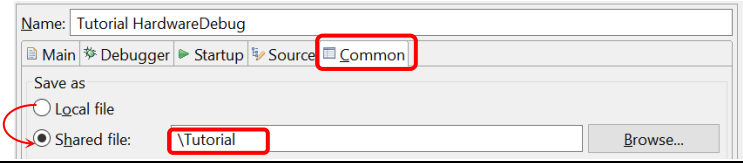


Figure 5-9 Attach Launch Configuration to Specific Project

5.3 Launch Bar

This section explains the usage of 'Launch Bar' in the toolbar area of the e² studio main window. This interface is hidden by default in some of e² studio versions.
The interface shown below builds and debugs the selected launch target. (May or may not be the same project as the active project in Project Explorer.)

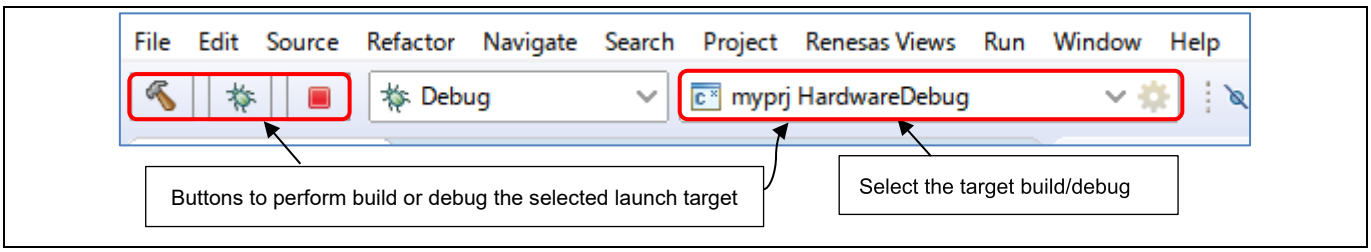


Figure 5-10 Launch Bar Interface

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 shown or 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 the 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 that exist in the [Debug] view:

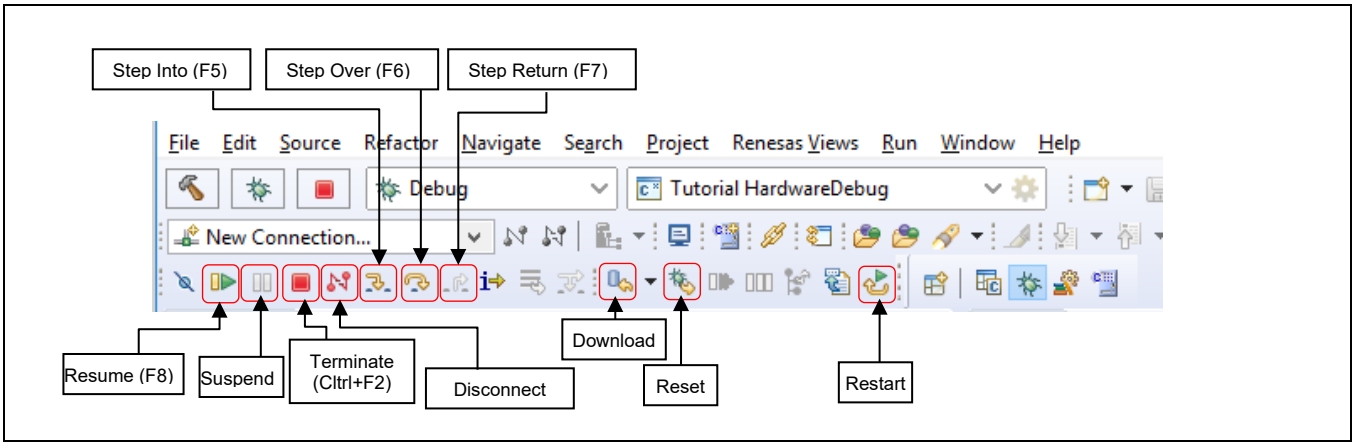


Figure 5-11 Useful Toolbars In Debug Views

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

The program can be paused by a breakpoint or by clicking the button. When the program is paused, users can perform the following operations:

- The button or [F5] can be used for stepping into the next method call at the currently executing line of code.
- The 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.
- The button can be clicked again to resume running.

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

The other operations are as follows:

- The button can be clicked to start a new debug session. This is the same operation as clicking on and then .
- The button can be clicked to reset the program to the entry point at PowerOn Reset.
- The button is used for re-downloading the binary file to the target system.

Note: To demonstrate the features in the following section, please use the sample code for RX64M from the Renesas website as instructed in section 3.4.1.

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 is executed. e² studio allows software and hardware breakpoints to be set explicitly in the IDE. Any breakpoints added via double-clicking 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 users 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 [Window] → [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).
   The breakpoint is restored to enabled by selecting [Enable Breakpoint] from the context menu. Alternatively, double-clicking while pressing the shift key switches between disabling and enabling of the breakpoint.
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.

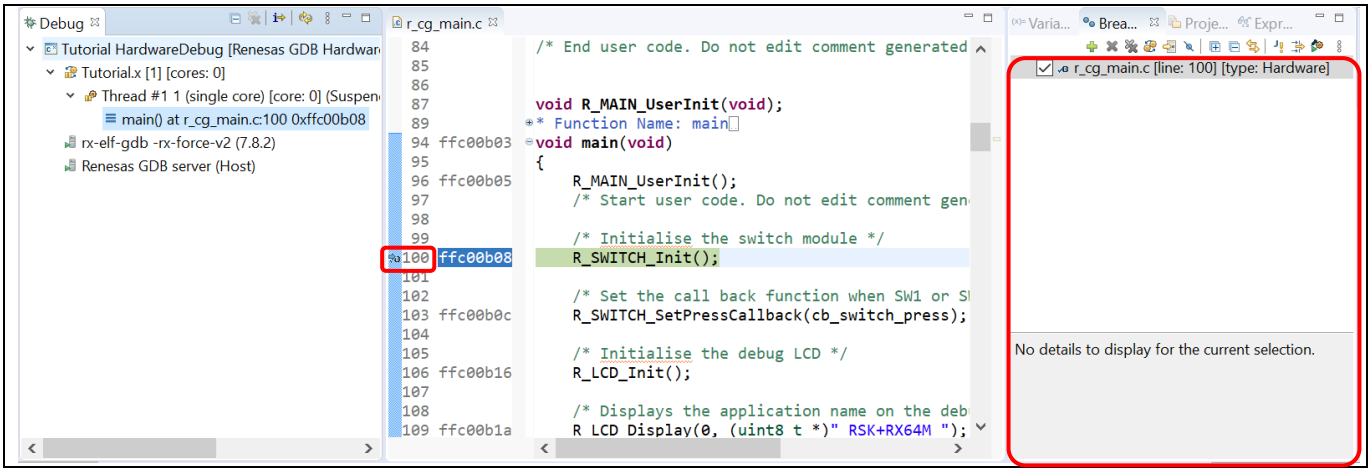


Figure 5-12 [Breakpoints] View

5.4.2 Expressions View

The 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. Values of variables can be registered by names, expressions (e.g. “Aval+Bval*2”), and cases of type casting (e.g. “(struct mystr *)&buf[1]”).

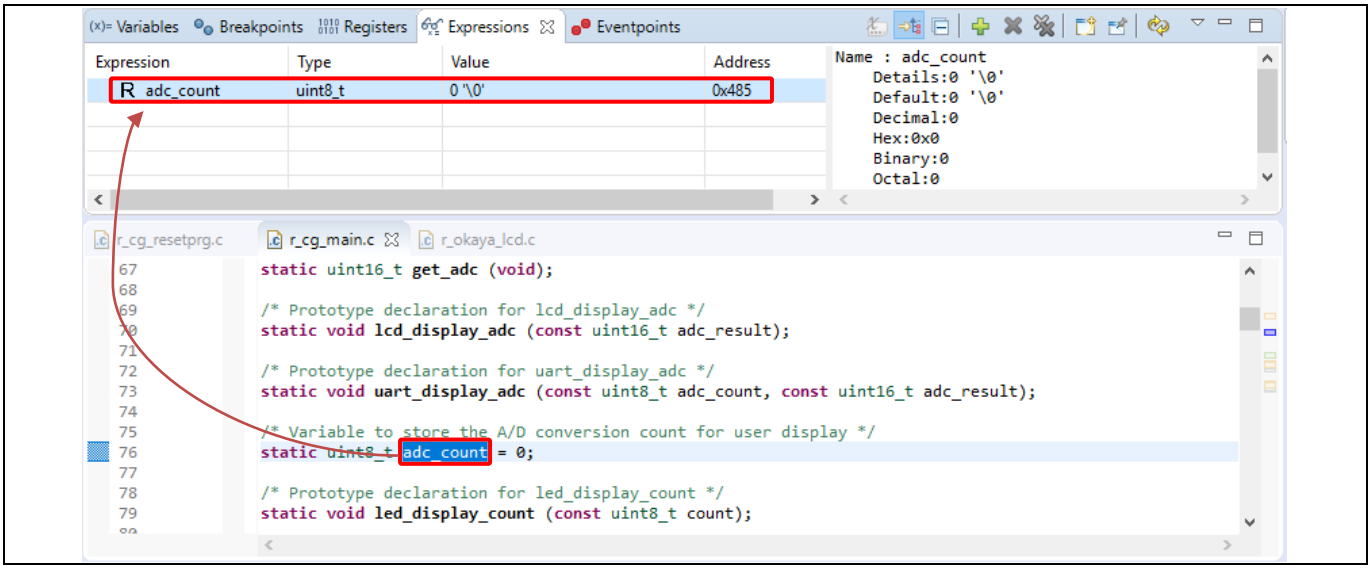


Figure 5-13 [Expressions] View

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 the [Expressions] view. (Alternatively, right-click at the global variable to select the “Add Watch Expression…” menu item to add it to the [Expressions] view).
3. In the [Expressions] view, right-click to select the “Real-time Refresh” menu item. This refreshes the expression value in real-time when the 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 the “Disable Real-time Refresh” menu item.

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

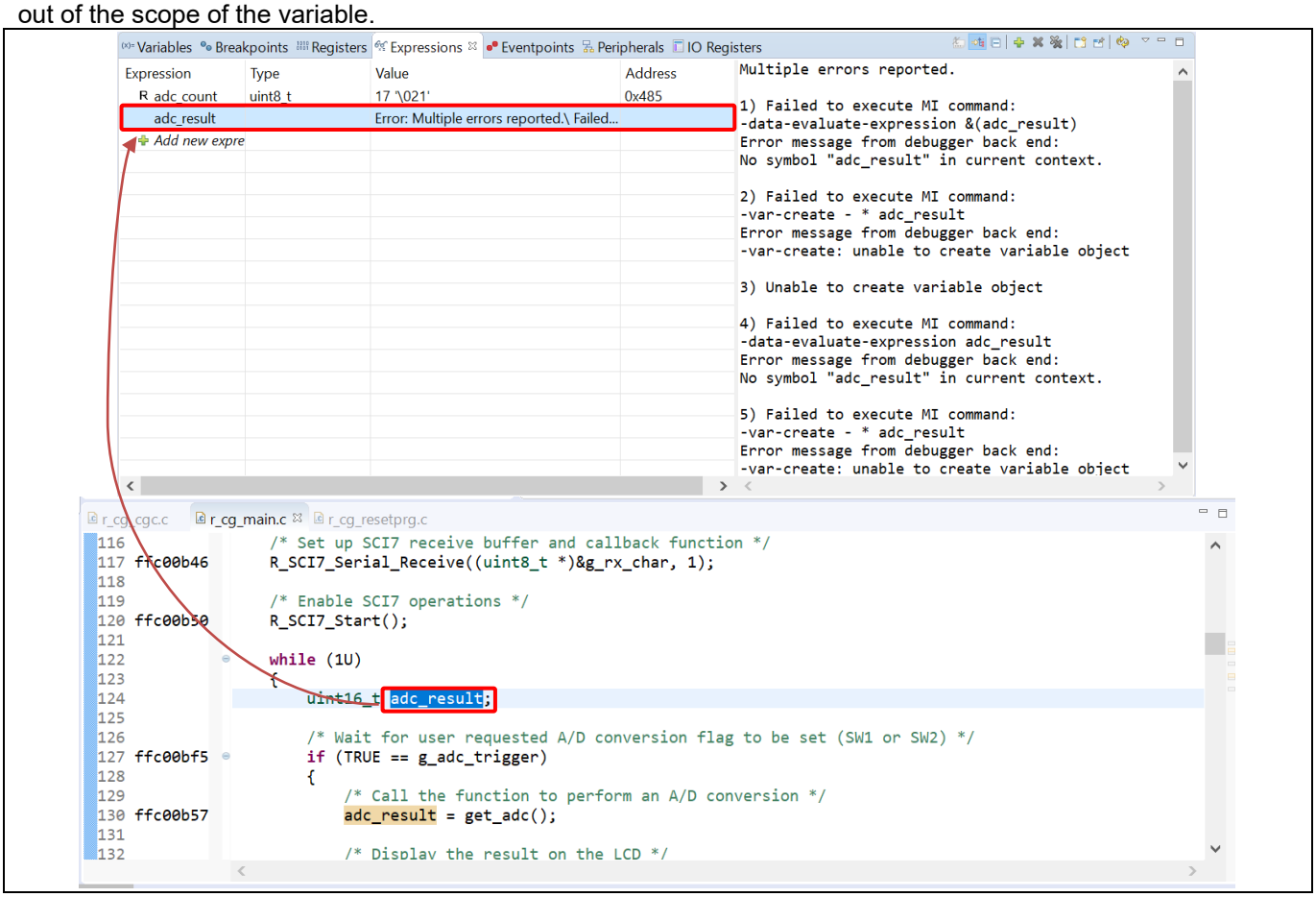


Figure 5-14 Add Local Variable to the Expressions View

For a variable of which address is uniquely determined (static variable), the value can be referenced by explicitly specifying the scope even when the variable is executing outside the scope.
For example, if you want to refer to the "myval" variable in the scope of the function myfunc(), you can create it in the Expressions view in the format "myfunc::myval" (with two colons between them).

5.4.3 Registers View

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

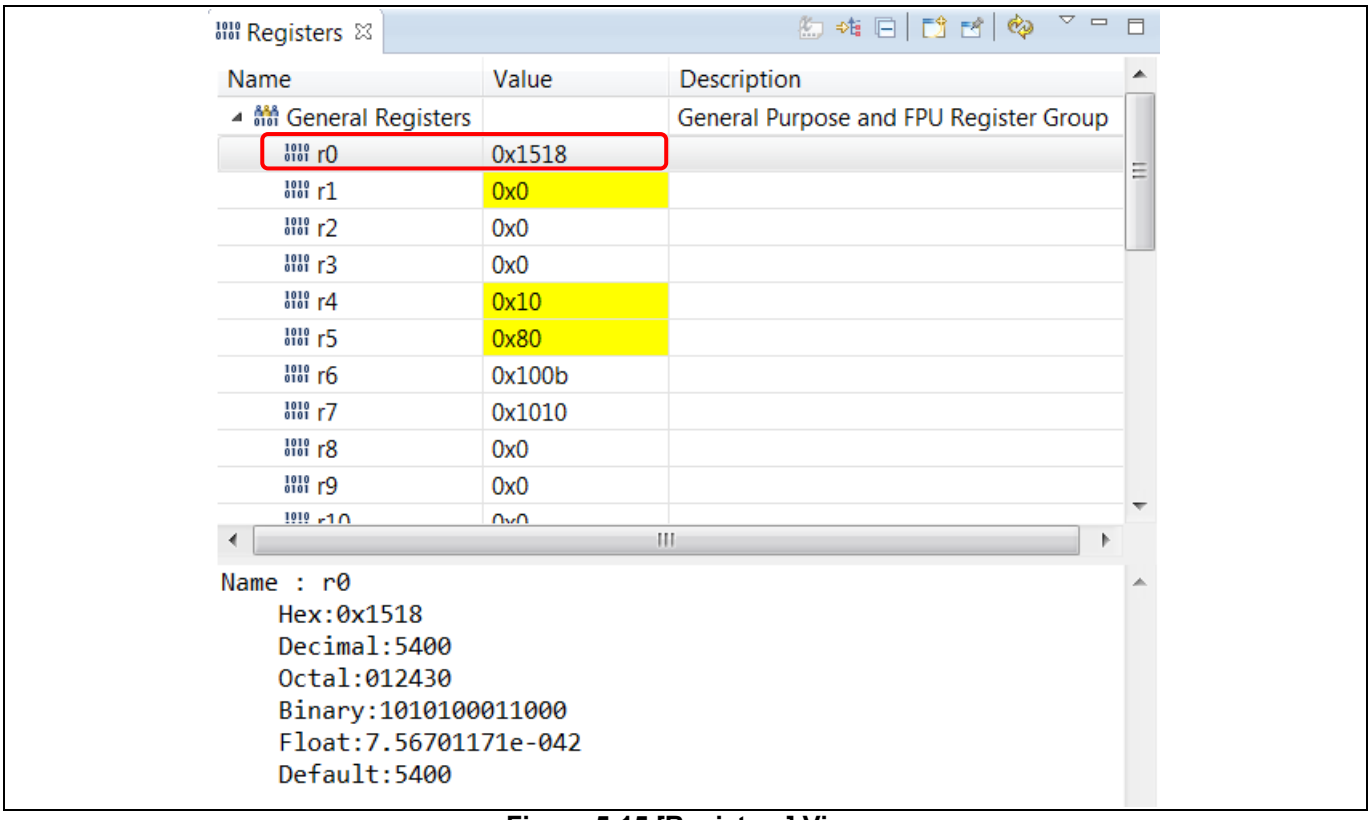


Figure 5-15 [Registers] View

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

The 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 a variable (e.g. “adc_count”) in the Memory view,

1. Click [Window] → [Show View] → [Memory] or icon to open the [Memory] view.
2. Click the icon to open the [Monitor Memory] dialog box. Enter the address of the variable “&adc_count”.

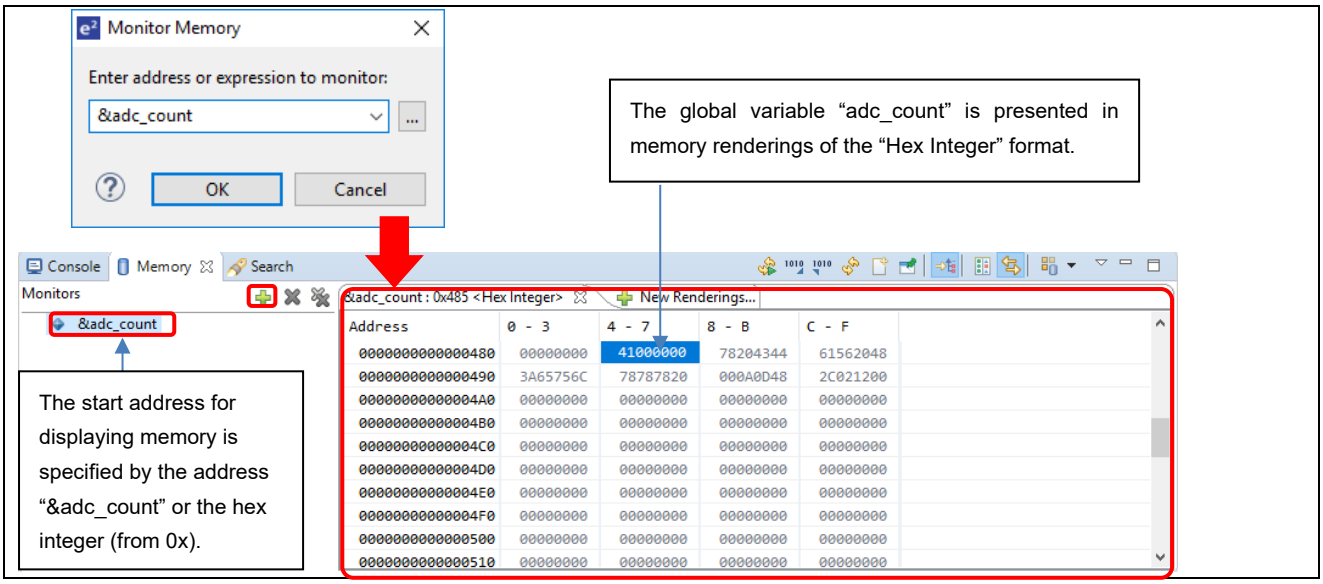
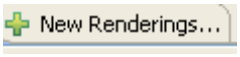


Figure 5-16 [Memory] View (1/2)

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

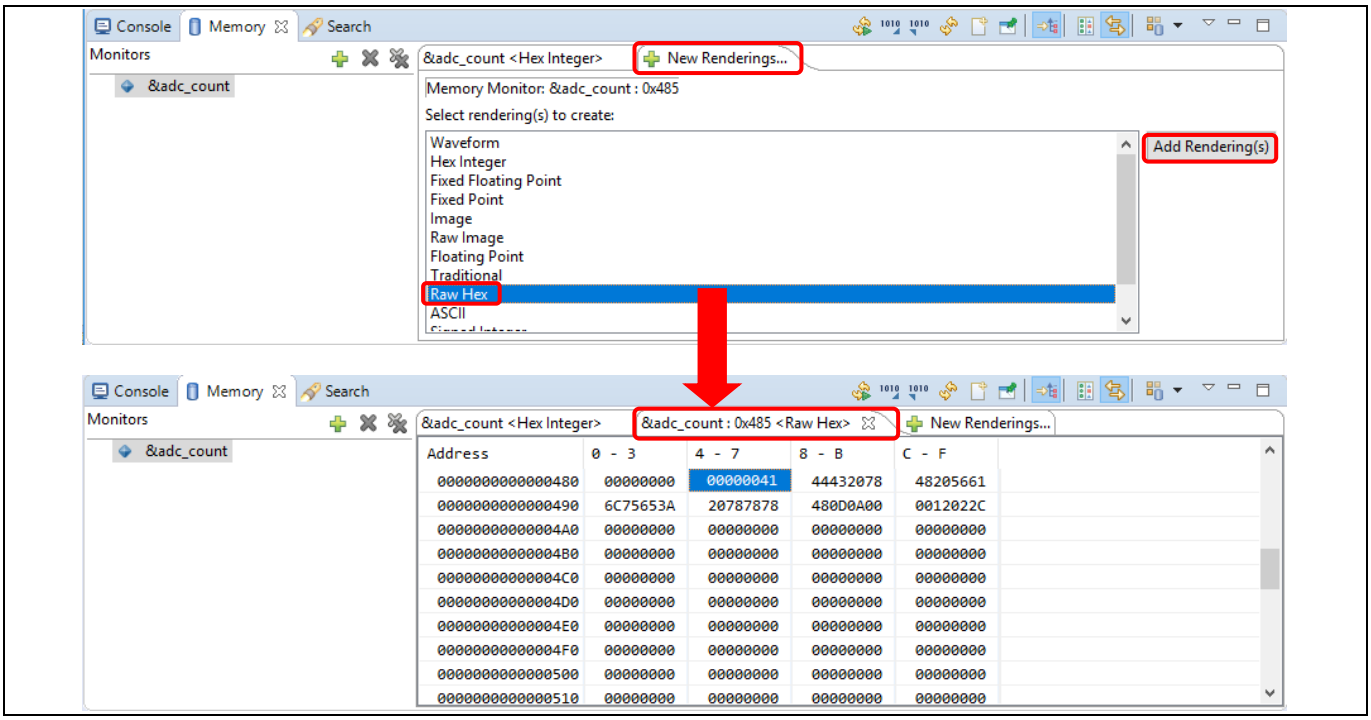


Figure 5-17 [Memory] View (2/2)

5.4.5 Disassembly View

The Disassembly view shows the loaded program as assembler instructions mixed with the source code for comparison. The current executing line is highlighted by an arrow marker in the view. In the [Disassembly] view, users 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.

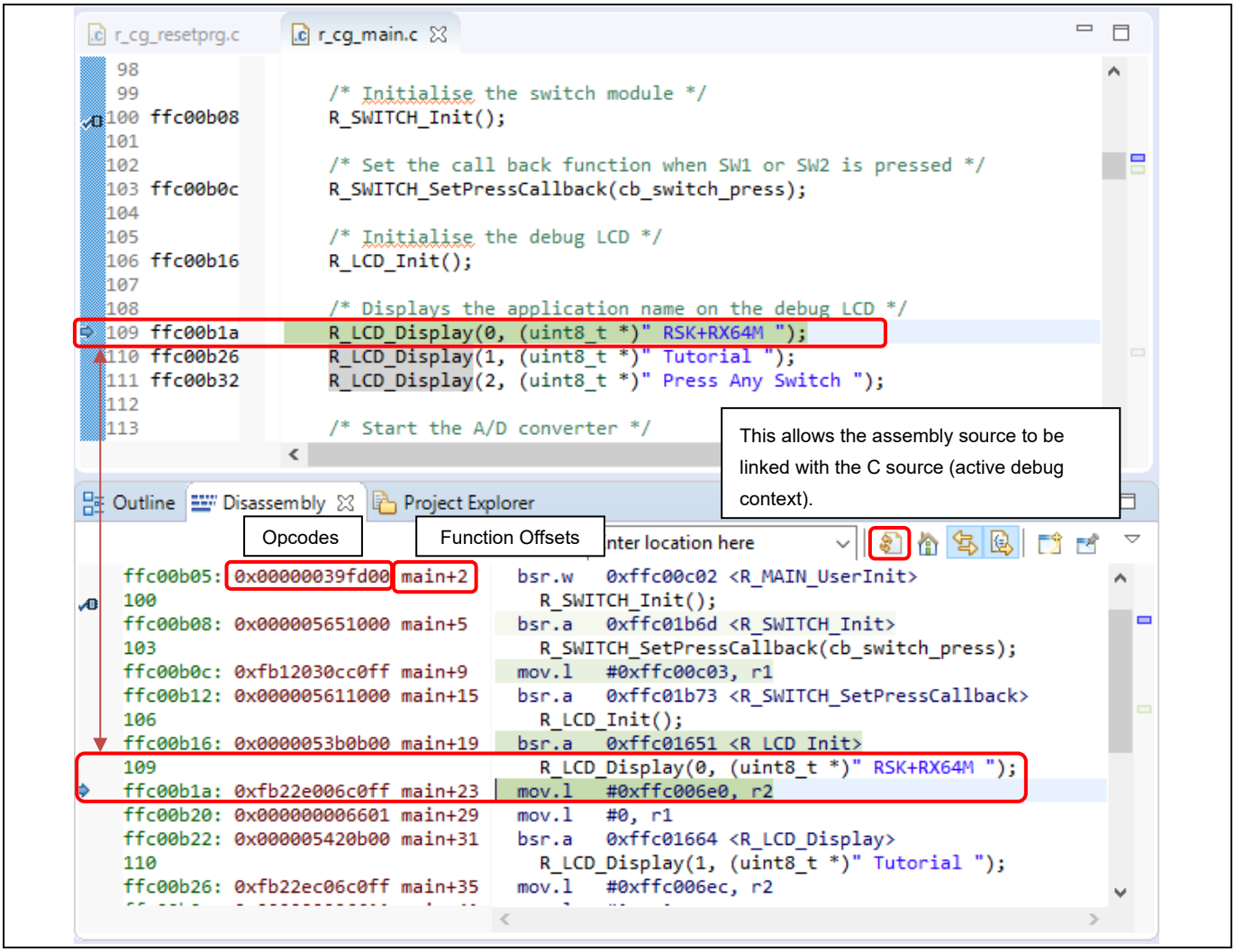


Figure 5-18 [Disassembly] View

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 the [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.

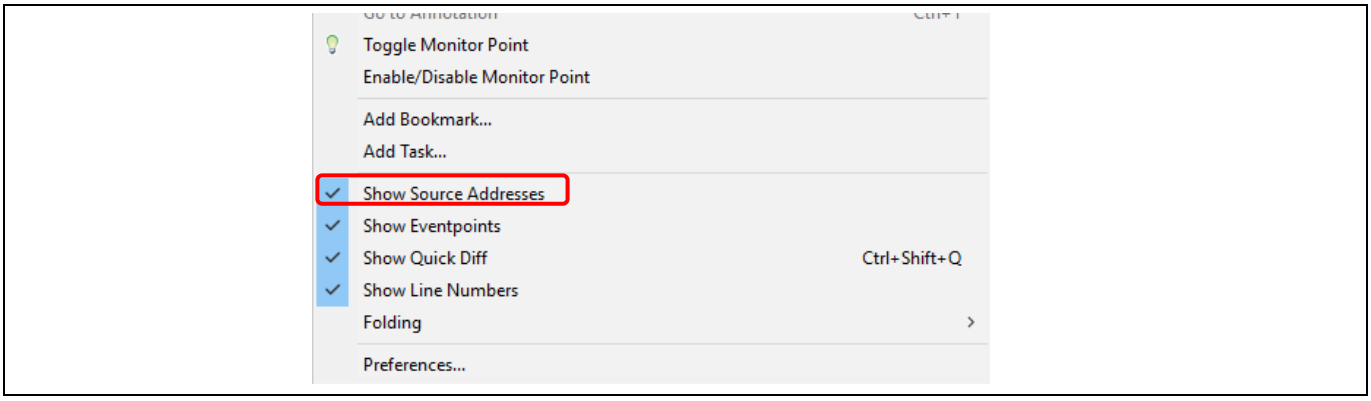


Figure 5-19 Source Addresses Menu

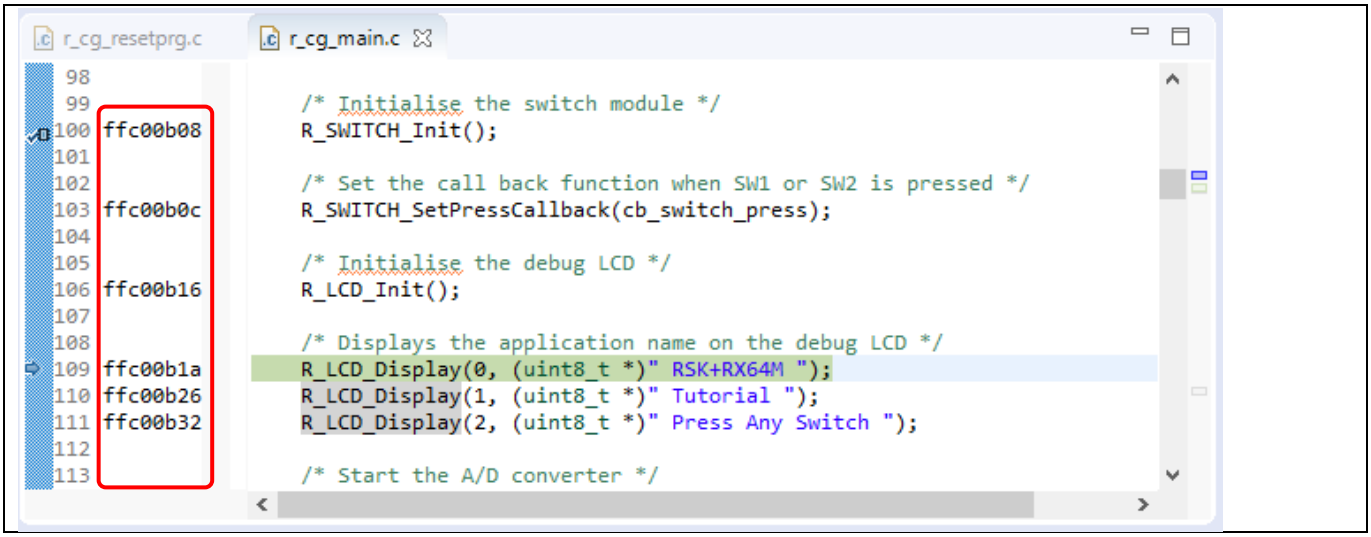


Figure 5-20 Source Addresses Displayed in the Editor

5.4.6 Variables View

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

Please refer to 'Expressions' view (refer to section 5.4.2) to watch global variables or external variables out of current program scope.

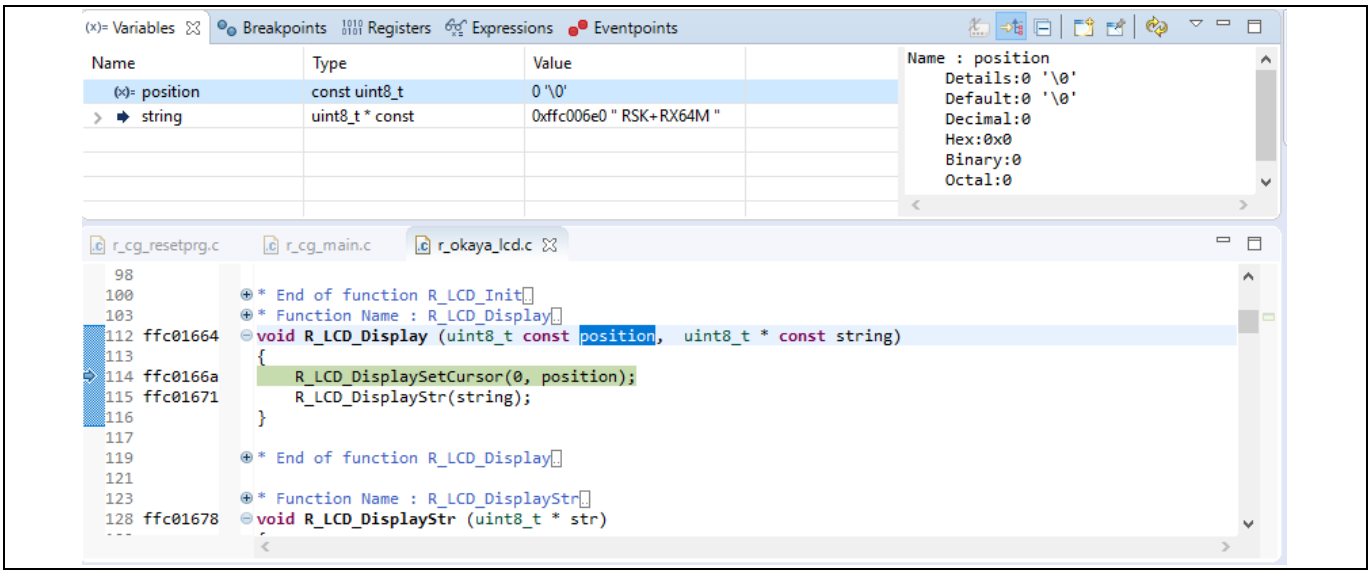


Figure 5-21 [Variables] View

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 temporarily allocated to accumulator registers may not appear in this view. Please refer to the Disassembly view if necessary to confirm which registers (also refer to the Registers view) or memory are used as the variable.
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. The [Eventpoints] view enables users to set up or view defined events of different category; e.g. trace start, trace stop, trace record, event break, before PC break, 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 events, a condition is satisfied immediately **before** execution of the instruction at the specified address) or other events (e.g. with events, a 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 users to set up complex address and data matching criteria.

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

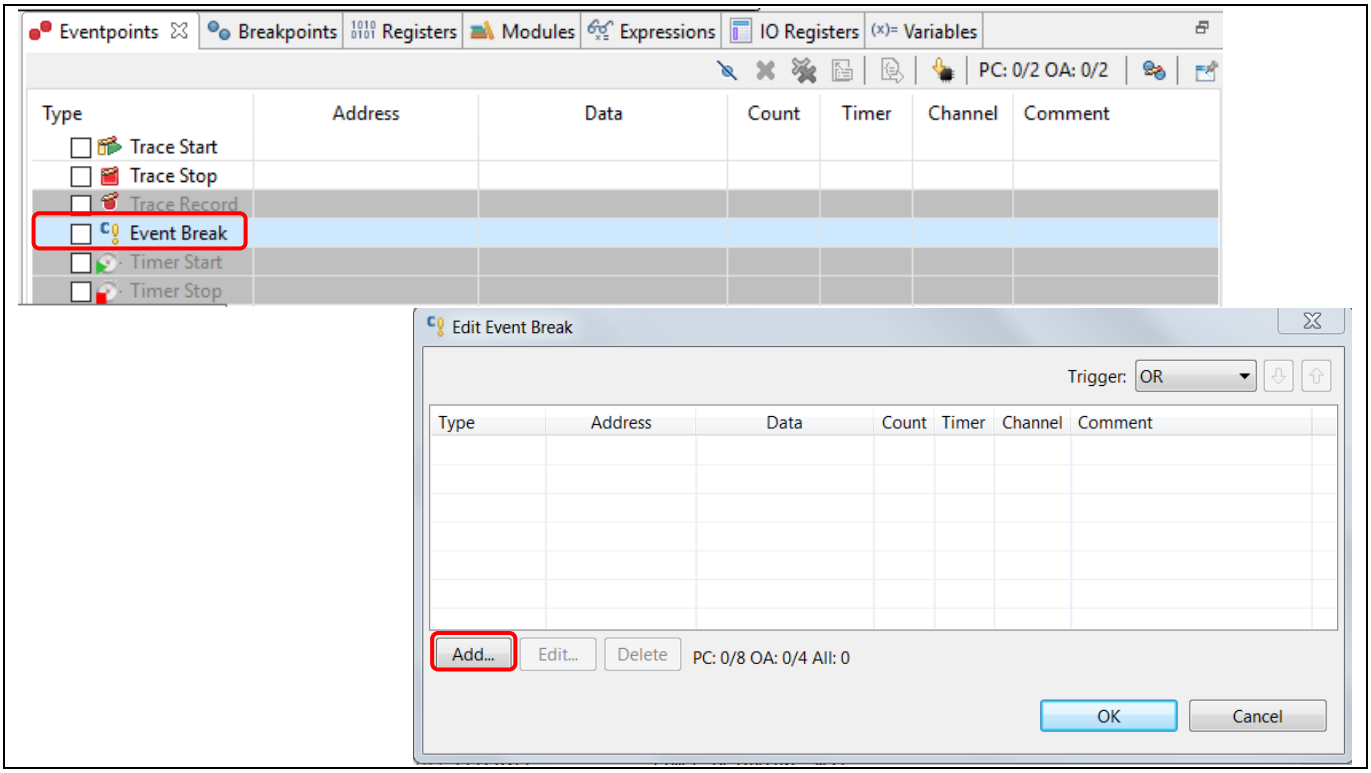


Figure 5-22 [Eventpoints] View (1/2)

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 the “Event Break” option to open the [Edit Event Break] dialog box.
3. Click the [Add…] button to continue.

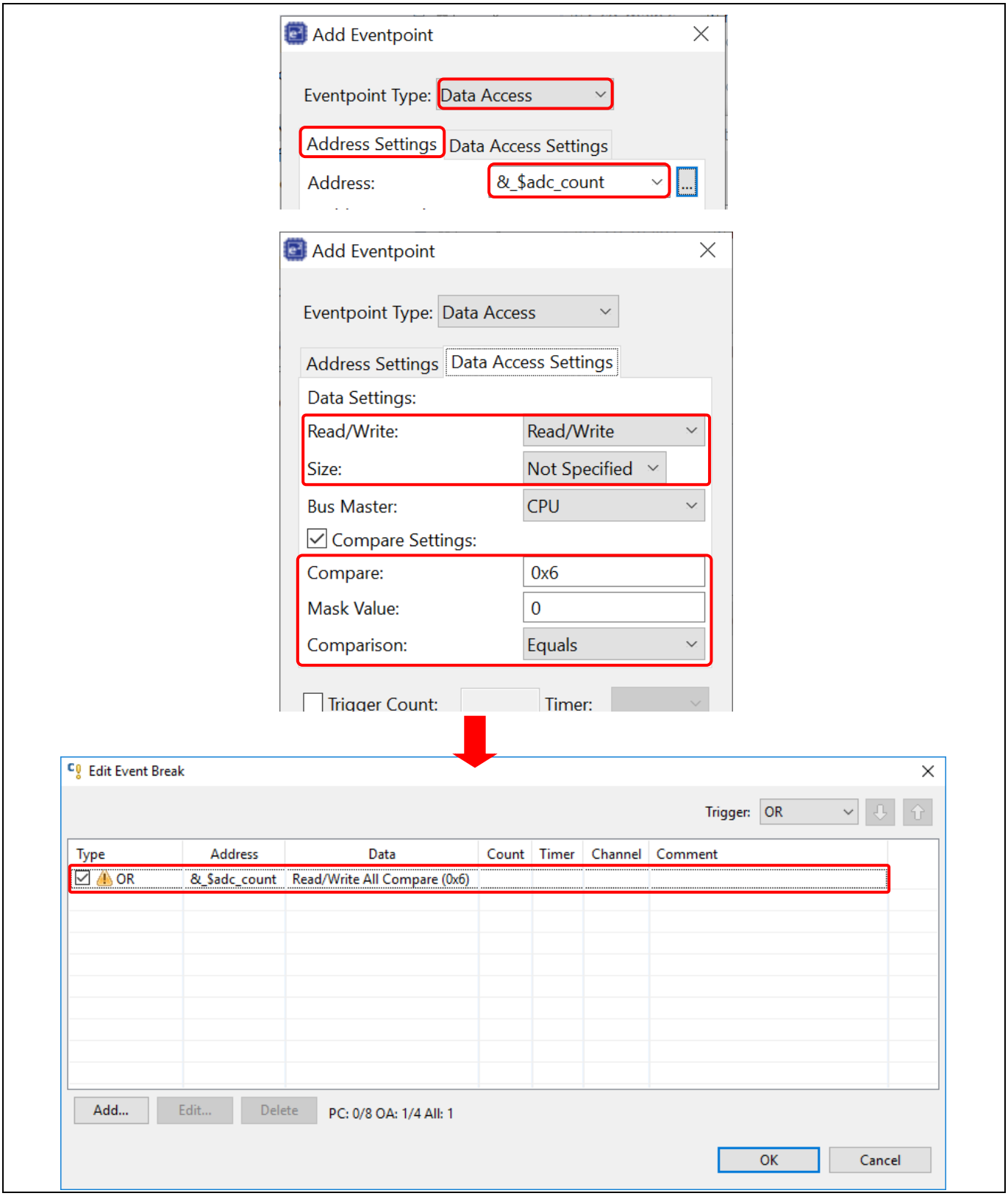


Figure 5-23 [Eventpoints] View (2/2)

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 equal 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 six times.

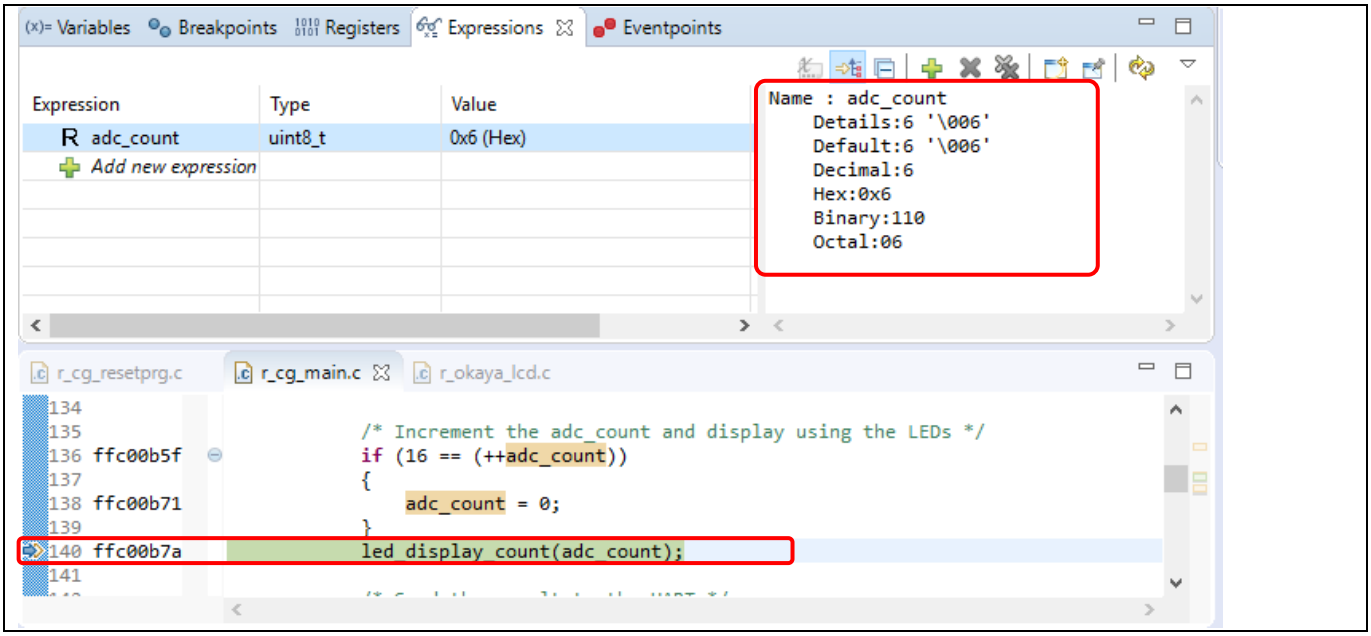


Figure 5-24 Execution of Event Break

Figure 5-24 shows that adc_count reaches the value of 6 (or 0x6) and 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 are also known as the Special Function Registers (SFR). The [IO Registers] view displays all the register sets defined in a target-specific IO file, including their names, addresses, and hex and binary values. Users can further customize their own [IO registers] view by adding IO registers selectively to the [Selected Registers] pane.

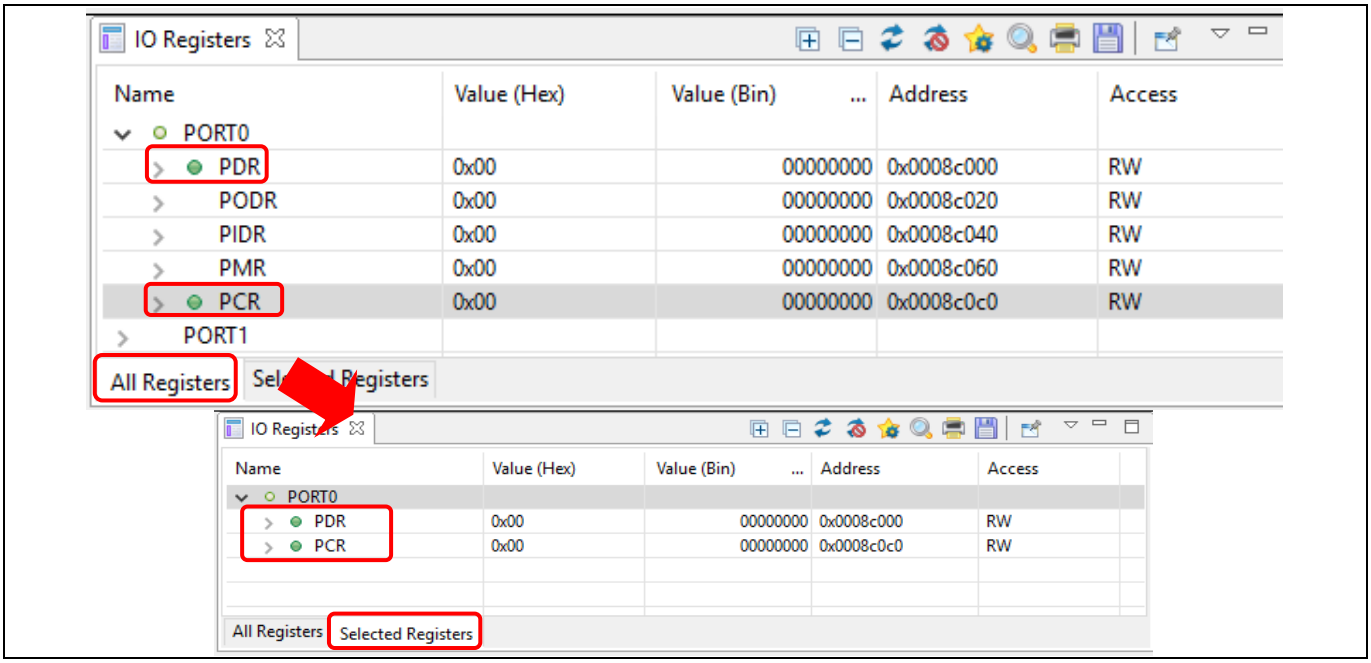


Figure 5-25 [IO Registers] View

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

1. Click [Windows] → [Show View] → [Others…]. In the “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 the 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 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 users 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), the oldest trace data is overwritten with the new data after the buffer has become full.

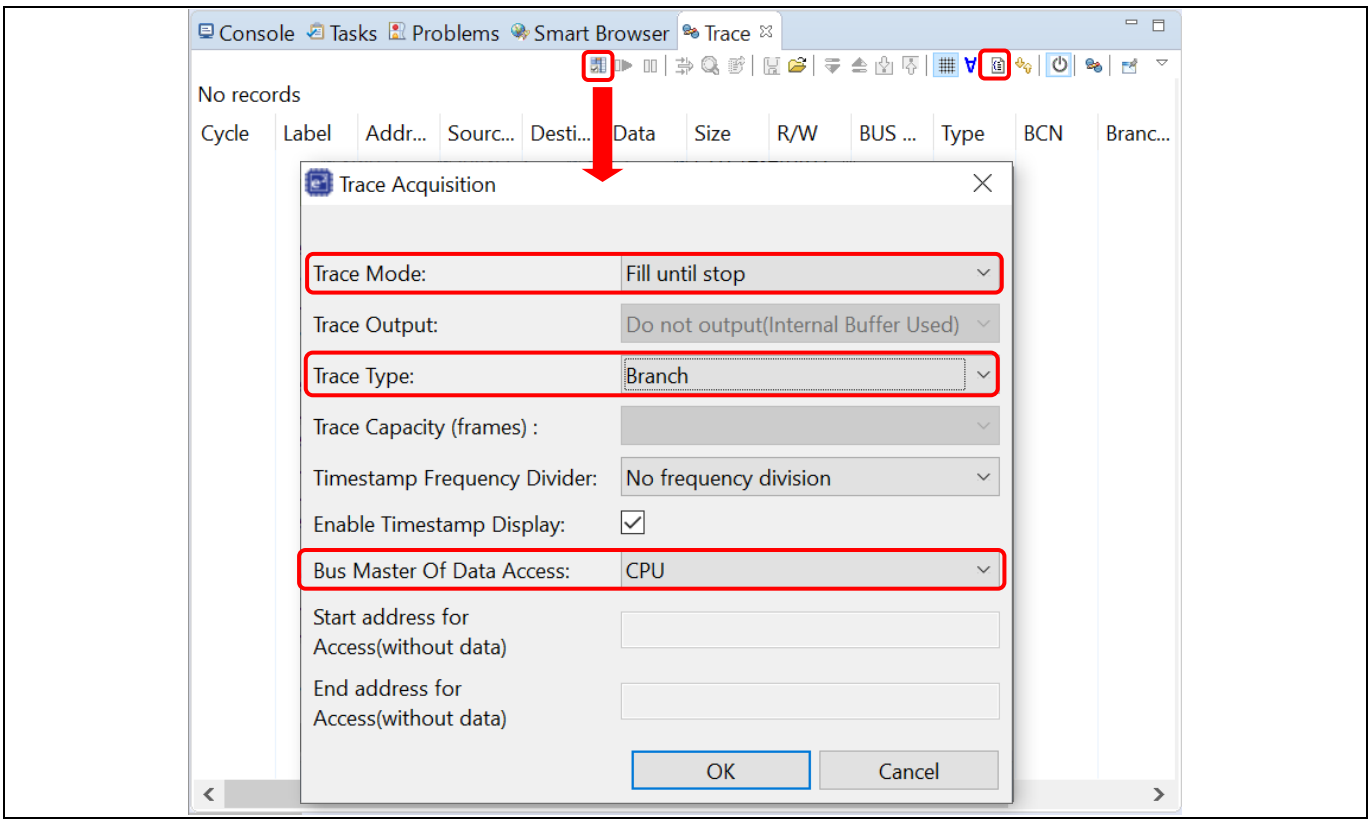


Figure 5-26 [Trace] View (1/2)

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

1. Click [Window] → [Show View] → [Others…]. In the “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.

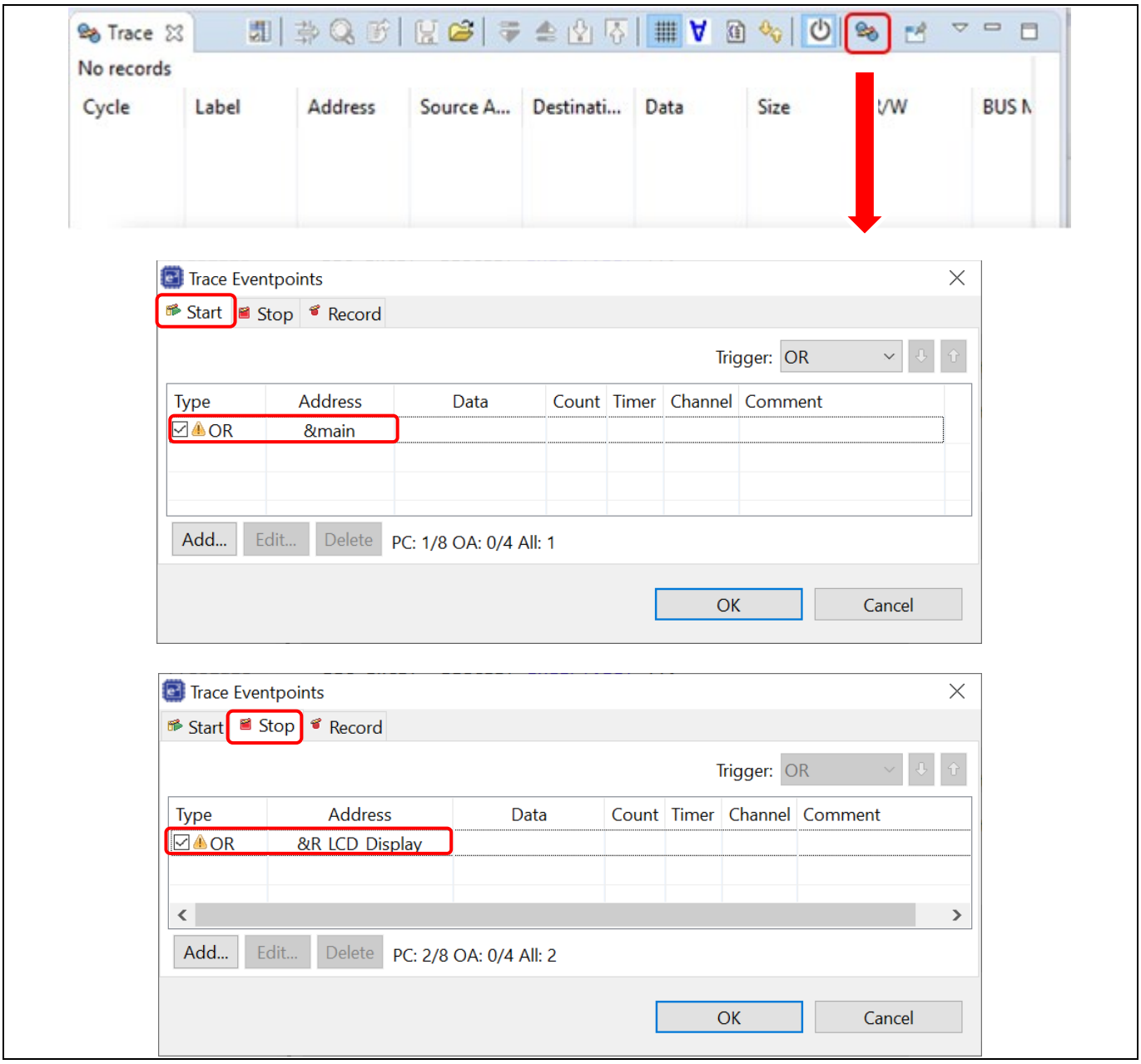


Figure 5-27 [Trace] View (2/2)

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

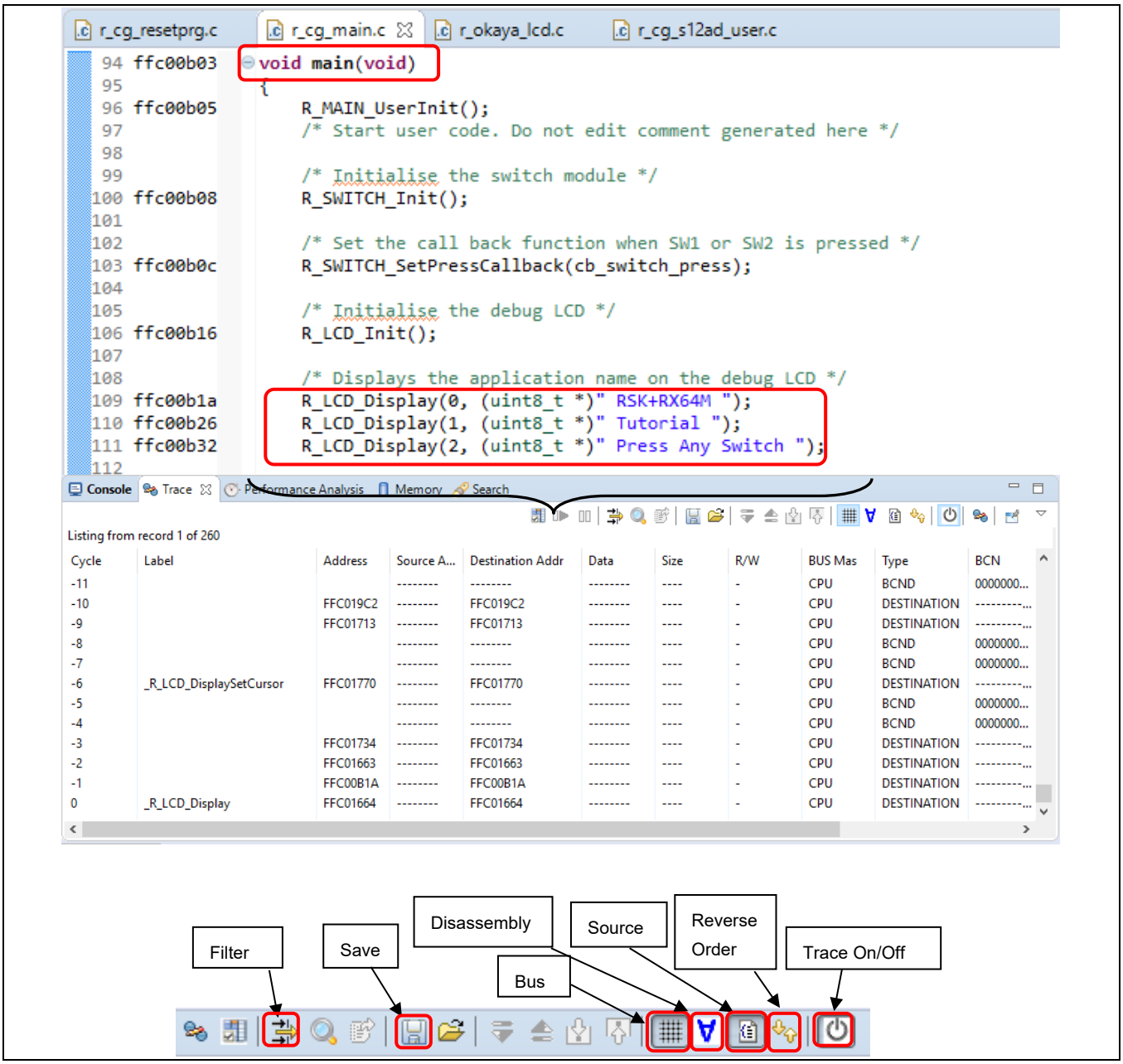


Figure 5-28 Point-To-Point Trace Between Two Functions

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: The external trace feature of RX device with the E20 emulator works only through Mictor-38-pin interface. However, it is not available through the 14-pin JTAG/FINE interface, even with the E20 emulator. The RX emulator interface specifications can be downloaded at the following site.

E1/E20/E2 Emulator, E2 Emulator Lite Additional Document for User's Manual (Notes on Connection of RX Devices)

https://www.renesas.com/search?keywords=R20UT0399

5.4.10 Memory Usage View

The Memory Usage view allows users to view the total memory size, usage of ROM and RAM ratio and detailed information of sections, objects, symbols or modules, vector tables, and cross references used in a 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 the Memory Usage view is different according to each kind of projects.
   a. The GUI of the Memory Usage view for executable project which uses Renesas Toolchain includes 3 regions: (1) Group size region, (2) RAM/ROM Usage region and Device Memory Usage region, (3) Detail table region. The map file location is shown at the bottom bar.

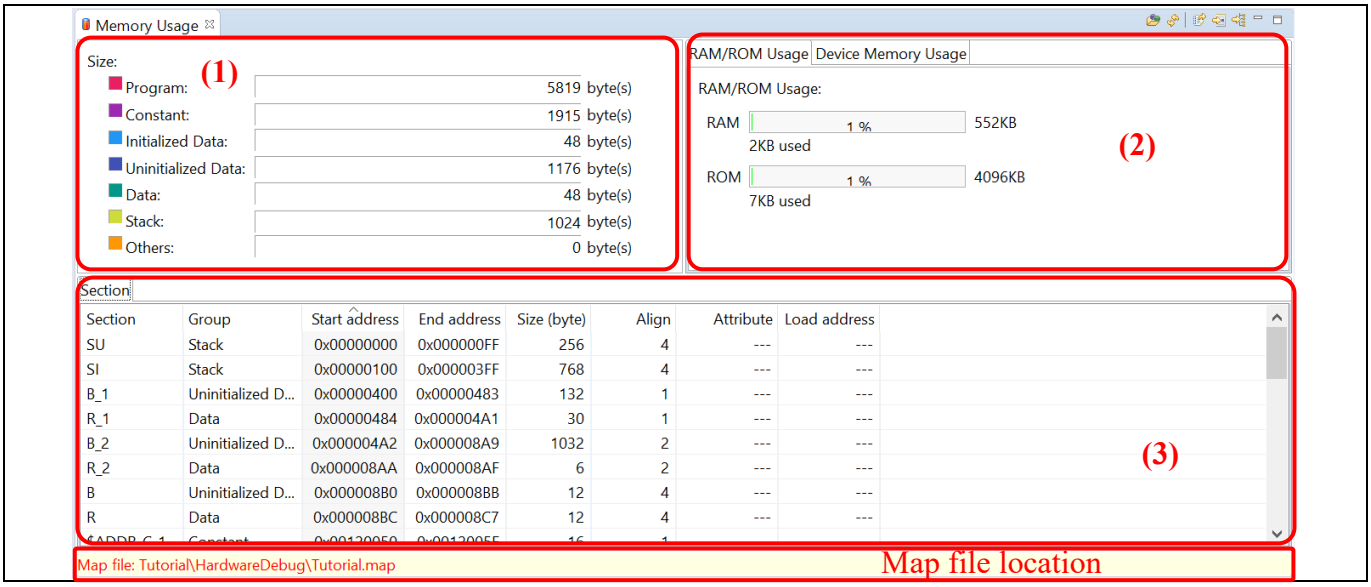


Figure 5-29 Executable Project - Renesas Toolchain

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

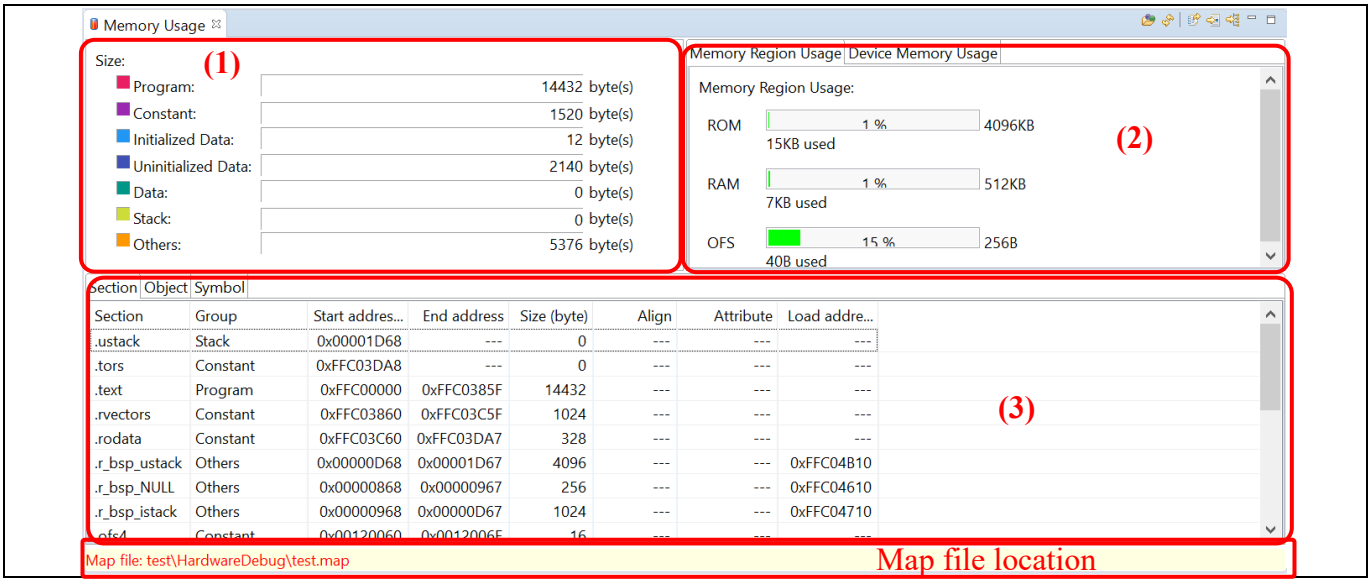


Figure 5-30 Executable Project - GCC Toolchain

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

Note: Only available for Renesas CC-RX, CC-RL, or CC-RH toolchains.

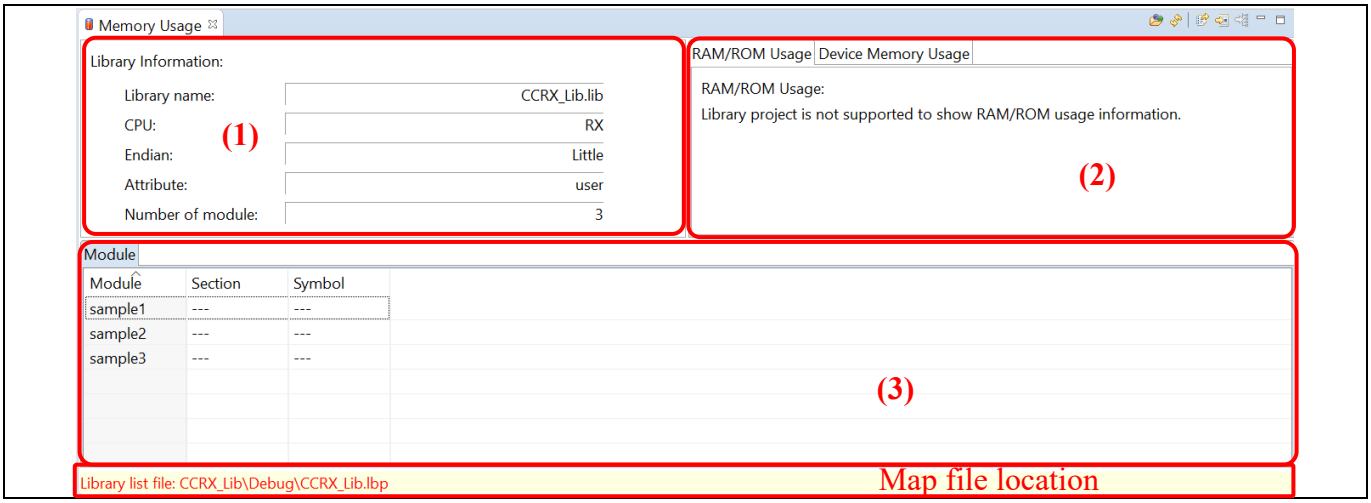


Figure 5-31 Library Project - Renesas Toolchain

d. The Memory Usage view is not available for the library project which uses GCC Toolchain.

Group Size view:

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

Note: This view only displays an 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 modules.

RAM/ROM Usage region:

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

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

Memory Region Usage region:

Displays the usage ratios for the address ranges of the memory region (memory block) of the linker script for a project that uses a GCC toolchain. The display feature of this region is similar to RAM/ROM usage region.

Device Memory Usage region:

Shows the device memory of selected project's device. Each memory area shows name, start address, end address, and the amounts of memory in use (in bytes and as percentages) relative to the whole size of the area.

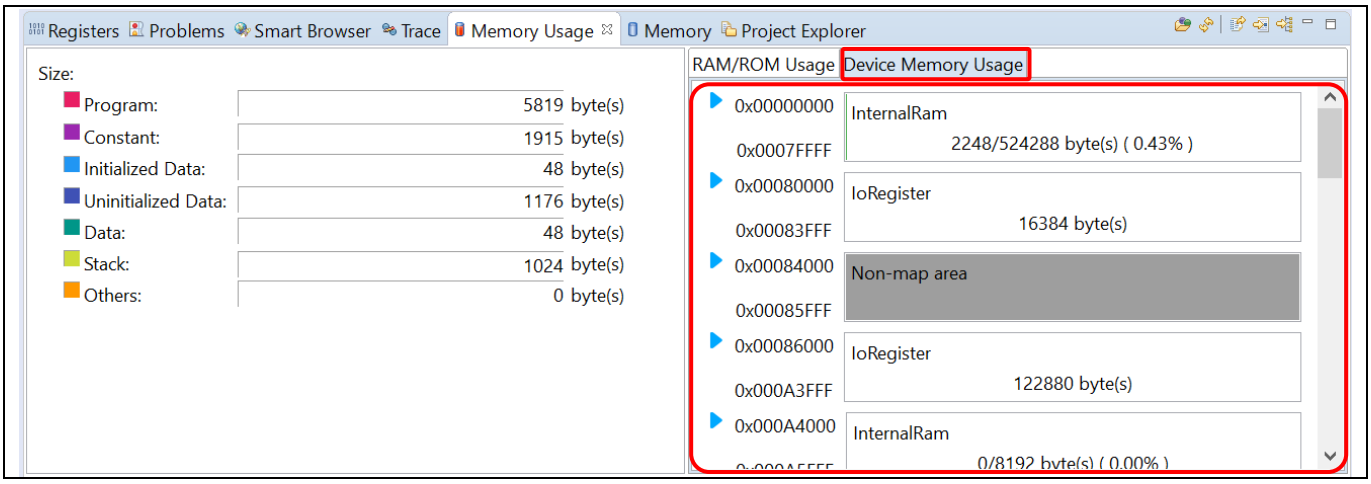


Figure 5-32 Device Memory Usage Region

Expand memory area to see all sections. The color of sections corresponds to that of Group Size Region.

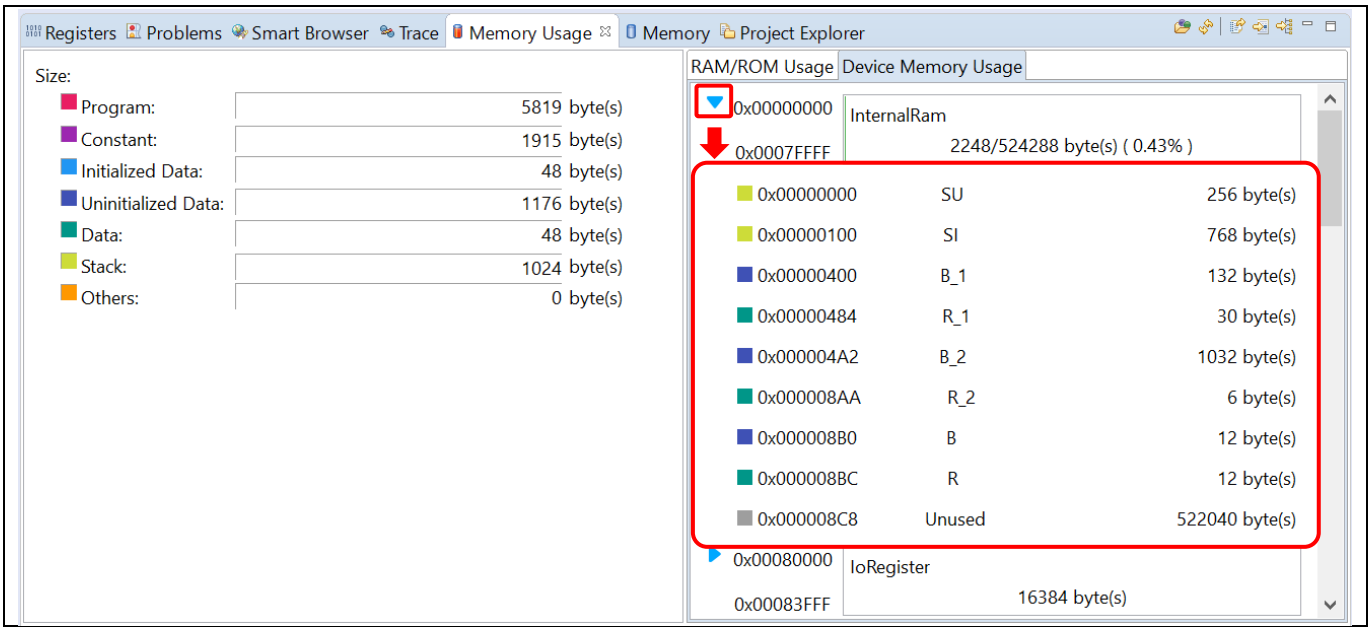


Figure 5-33 Expand Memory Area

Detail table region:

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

- “Section” tab: Contains the “Linkage map” table which displays the list of Sections analyzed from the map file and its detailed information.
- “Object” tab: Contains the “Object” table which displays the list of Objects analyzed from the map file and its detailed information.
- “Symbol” tab: Contains the “Symbol” table which displays the list of Symbols analyzed from the map file and its detailed information.
- “Vector” tab: Displays the vector table information that is retrieved from the map file. This tab is only available for an executable project that is configured to work with Renesas CC-RX/CC-RL/CC-RH toolchains.
- “Cross Reference” tab: Displays the cross reference information that is retrieved from the map file. This tab is only available for an executable project.
- “Module” tab: Contains the “Module” table. This tab is only available for a library project that is configured to work with Renesas CC-RX/CC-RL/CC-RH toolchains.

Map file location:

Displays the information of the map file (*.map) or library list file (*.lbp) from a project. Users can see the relative path of the selected map file or library list file at the bottom of the 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 the [Help] tab to open the Help menu.

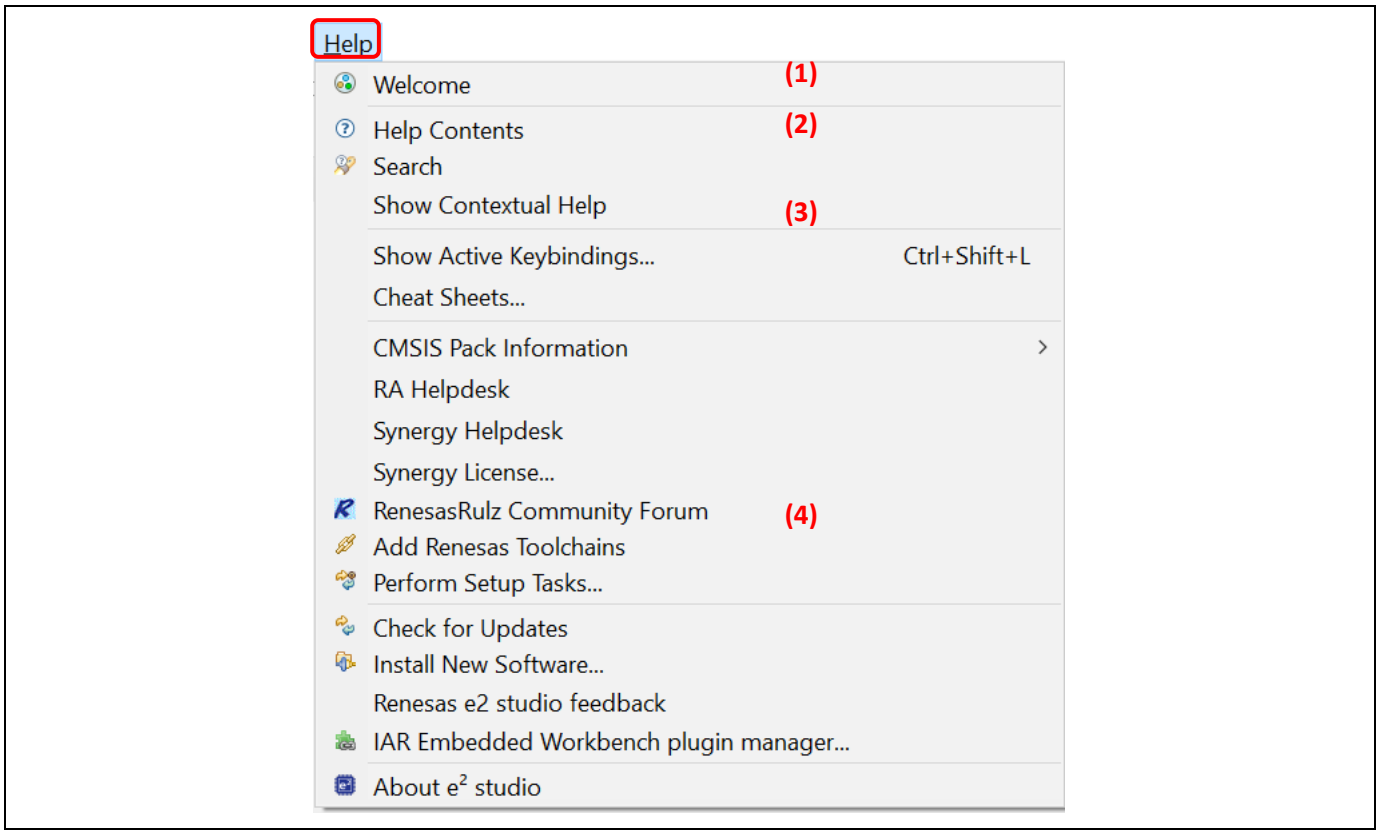


Figure 6-1 Help – Help Menu

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 [Help Contents]. For example, the “Debugging Projects” topic 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 to an online forum that is dedicated to topics and discussions related to the e² studio (Internet connection is required).

Revision History	e² studio User's Manual: Quick Start Guide

Rev.	Date	Description	
		Page	Summary
1.00	Jun.21.23	-	First edition for RX, RL78, RH850 based on e² studio 2023-04 issued.
1.10	Mar.26.24	-	Updated installation procedure as the latest e² studio version. Added descriptions of RISC-V MCU.
1.20	Feb.28.25	-	Updated the version numbers for supported operating systems. Added statements regarding DA devices.

e² studio User's Manual: Quick Start Guide

Publication Date: Rev.1.20 Feb.28.25

Published by: Renesas Electronics Corporation

e² studio
User's Manual:
Quick Start Guide
RX Family
RL78 Family
RH850 Family
RISC-V MCU
DA Devices

Renesas Electronics Corporation