User's Manual

16

e^2 studio Integrated Development Environment

User's Manual: Getting Started Guide

RENESAS MCU
RZ Family

Renesas Electronics
www.renesas.com

Rev.1.02 Sep.2020

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 for developing applications and systems and provides an outline of its features .

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

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

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

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

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

CHAPTER 1 GENERAL
CHAPTER 2 INSTALLATION
CHAPTER 3 IMPORT / GENERATE PROJECT
CHAPTER 4 BUILD
CHAPTER 5 DEBUG
CHAPTER 6 HELP

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

Conventions

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

List of Abbreviations and Acronyms

Abbreviation	Full Form
API	Application Programming Interface
ARM	Advanced RISC Machine
CAN	Controller Area Network
CDT	C/C++ Development Tooling
CG	peripheral Code Generator
CPU	Central Processing Unit
ELF/DWARF	Executable and Linkable Format / Debugging With Attributed Record Formats
FIT	Firmware Integration Technology
GB	Gigabyte
GCC	GNU Compiler Collection
GDB	GNU project DeBugger
GHz	Giga Hertz
GUI	Graphical User Interface
IDE	Integrated Development Environment
I/O	Input / Output
JTAG	Joint Test Action Group
LCD	Liquid Crystal Display
LED	Light Emitting Diode
LVDS	Low-Voltage Differential Signalling
MCU	Microcontroller Unit
OS	Operating System
PC	Personal Computer or Program Counter
PMOD	Peripheral MODule (an open standard defined by Digilent Inc^{TM})
RTOS	Real Time Operating System
SC	Smart Configurator
SD	Secure Digital
SFR	Special Function Registers
SVN	SubVersioN
TFT	Thin Film Transistor
URL	Uniform Resource Locator
USB	Universal Serial Bus

Table of Contents

1. General

Renesas e^2 studio is the Integrated Development Environment for Renesas embedded microcontrollers. e^2 studio is based on the industry-standard open-source Eclipse IDE framework and the C/C++ Development Tooling (CDT) project, covering build (editor, compiler, and linker control) and debug phases with an extended GNU Debug (GDB) interface support.

This chapter describes the system configuration and operating environment for the e^2 studio IDE for developing applications for the RZ family high-end ARM-based microprocessors.

1.1 System Configuration

Below is an example of a typical system configuration.

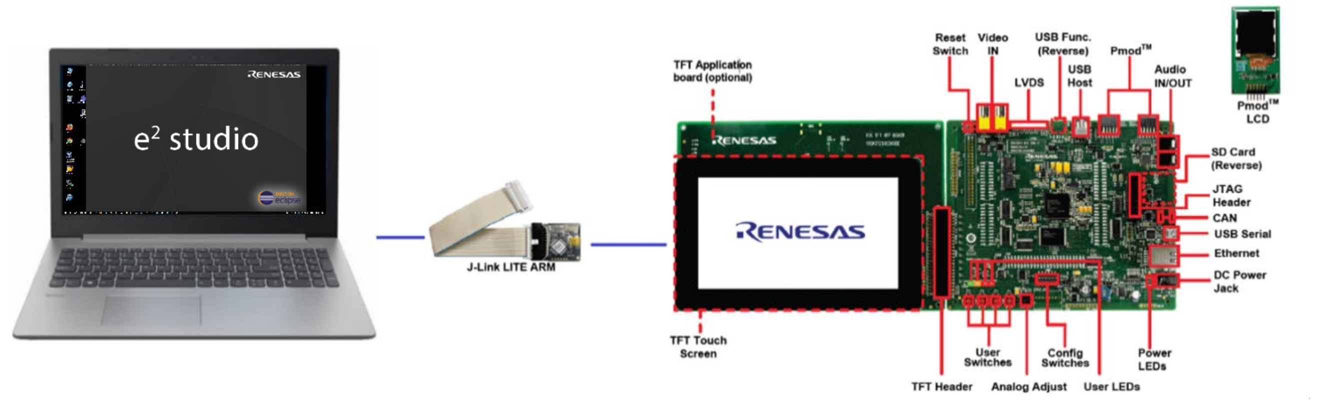


Figure 1-1. System Configuration Example - RZ/A1H Renesas Starter Kit

1.2 System Requirements

Below are the system requirements for this product.

Hardware Environment:

• Processor:	At least 1GHz (hyper-threading/multi-core supporting CPU)
• Main Memory:	At least 4GB of free space. 8GB is recommended.
• 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.

Software Environment:

Windows® 8.1 (32/ 64-bit versions) and Windows® 10 (32/64-bit versions)
64-bit OS is required for e^2 studio 2020-04 or later.

1.3 Supported Compiler

RZ family devices use GNU compilers for ARM architecture. Each device has a recommended version of the GNU compiler. Refer to https://www.renesas.com/rz. These compilers can be installed during the installation of e^2 studio by selecting them in the **Additional Software** pane (an Internet connection is required - see chapter 2.1) or can be downloaded from the web page below and installed separately (toolchain integration will be necessary).

- GNU ARM Embedded toolchain (GCC 6; 6-2016-q4-major or higher version):
  https://developer.arm.com/open-source/gnu-toolchain/gnu-rm/downloads
- GNU ARM Embedded toolchain (GCC^{note} 5; 5-2016-q3-update or lower version):
  https://launchpad.net/gcc-arm-embedded/+download
- GNUARM-NONE toolchain:
  https://gcc-renesas.com/rz/rz-download-toolchains/

Refer to chapter 2.2 for compiler installation and chapter 2.6 for e^2 studio toolchain integration.

Note: 'GCC' stands for 'the GNU Compiler Collection' (Refer to https://gcc.gnu.org/).

1.4 Additional Tool for Compiler Update

Libgen Update for GNU ARM Embedded Toolchains is a tool used to build 'newlib' specific libraries which is a C standard library implementation intended for embedded systems. Instead of a 'prebuilt' version of a library (a library built with a set configuration when the toolchain is created), the Libgen Update allows users to have their own 'project-built' library build configuration, using a custom set of options.

Libgen Update for GNU ARM Embedded Toolchains can be installed during the installation of e^2 studio if selected (an Internet connection is required), or can be downloaded from the web page below:

- Libgen for GNU ARM embedded toolchain (V1.2018.02 or higher version)
  https://gcc-renesas.com/rz/rz-download-toolchains/

Refer to chapter 2.2 for the installation of **Libgen Update for GNU ARM Embedded Toolchains**.

1.5 Supported Emulator

Segger J-Link Lite (RZ) and other emulators

For more detail, see RZ family product web page: https://www.renesas.com/rz

2. Installation

The latest e^2 studio IDE installer package can be downloaded from Renesas website for free. Renesas now supports both 32-bit and 64-bit versions of e^2 studio.

Please check detailed information from: https://www.renesas.com/e2studio . Note that user has to login to the Renesas account (in MyRenesas page) for the software download.

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

e^2 studio installer can be used to upgrade e^2 studio as well as new installation. However, it does not support update between major versions such as from v5.x to v6.y, or from v6.m to v7.n.

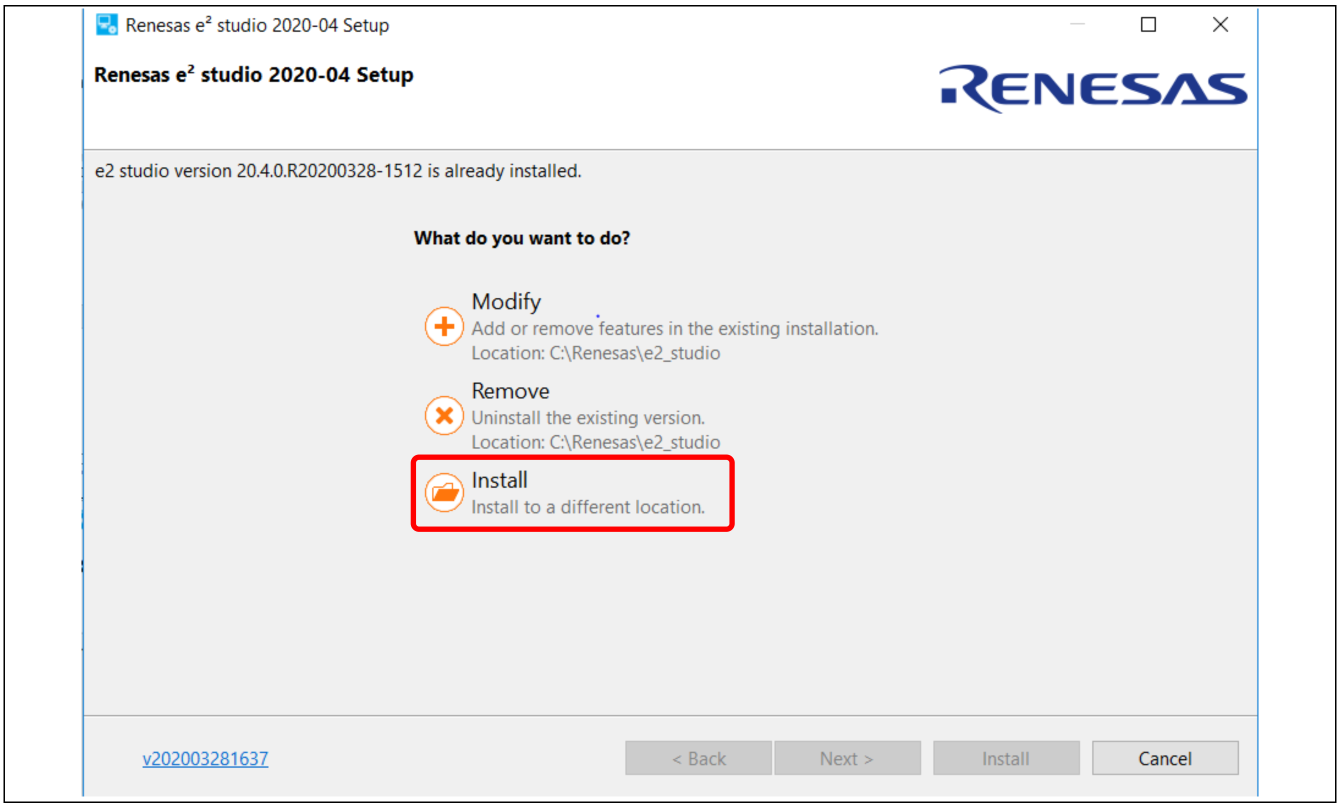
Please uninstall the earlier versions before installation. Alternatively, install new e^2 studio into a new folder if you would like to keep earlier versions.

The detailed information is described below.

2.1 Installation of e^2 studio IDE (64-bit version)

1. Double-click on e^2 studio installer to invoke the e^2 studio installation wizard page. Click [Install].

   Note: If e^2 studio was installed in your PC, the option to modify, remove the existing version or install e^2 studio to a different location will be displayed.


Renesas e^2 studio 2020-04 Setup

Renesas e^2 studio 2020-04 Setup

e2 studio version 20.4.0.R20200328-1512 is already installed.

What do you want to do?

Modify
Add or remove features in the existing installation.
Location: C:\Renesas\e2_studio

Remove
Uninstall the existing version.
Location: C:\Renesas\e2_studio

Install
Install to a different location.

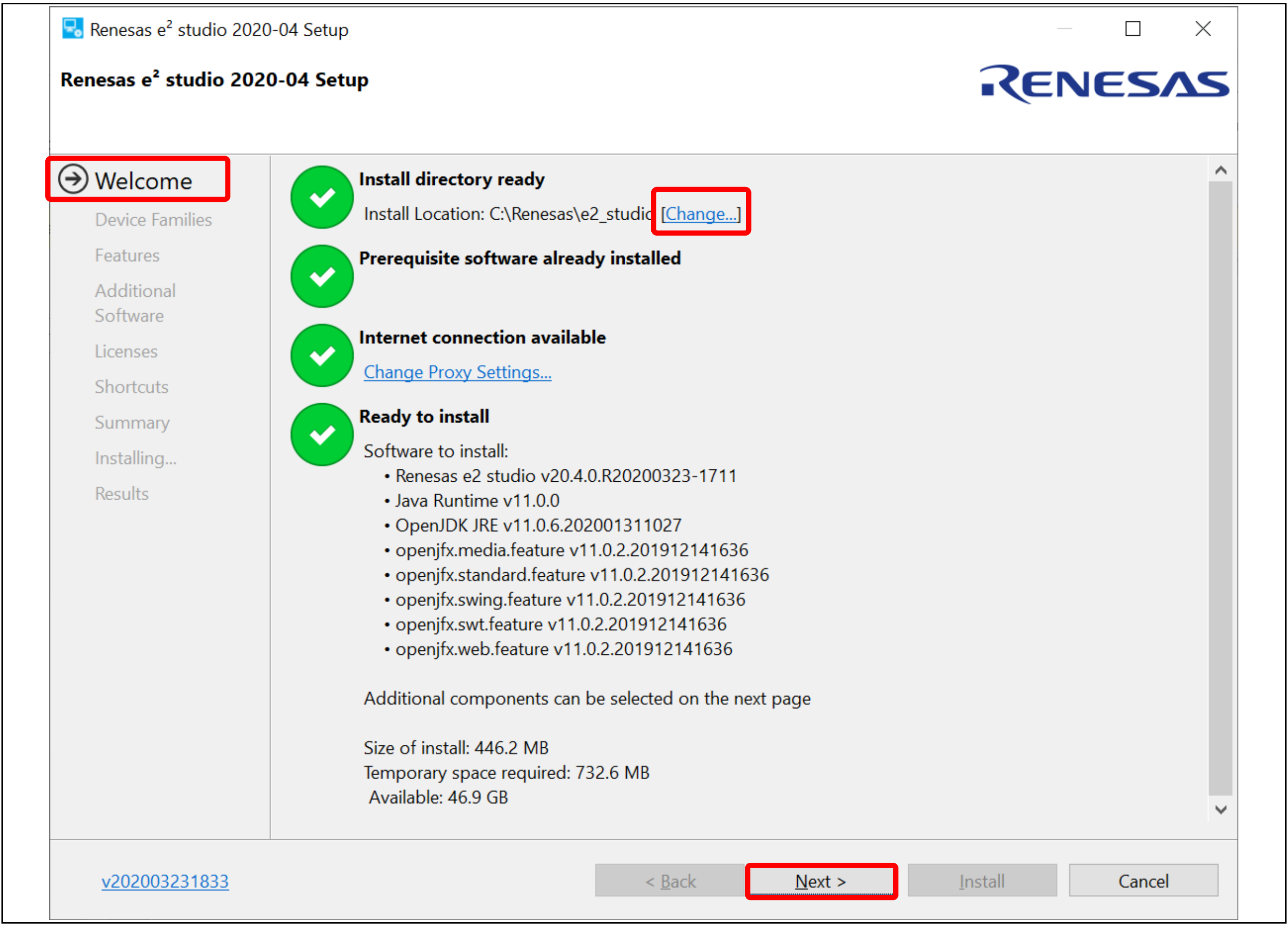
v202003281637

Figure 2-1. 64-bit e2 studio installation wizard

2. Welcome page

User can change the install folder by clicking [Change…]. Click [Next] to continue.

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



Install directory ready
Install Location: C:\Renesas\e2_studio [Change...]

Prerequisite software already installed

Internet connection available
Change Proxy Settings...

Ready to install
Software to install:
- Renesas e2 studio v20.4.0.R20200323-1711
- Java Runtime v11.0.0
- OpenJDK JRE v11.0.6.202001311027
- openjfx.media.feature v11.0.2.201912141636
- openjfx.standard.feature v11.0.2.201912141636
- openjfx.swing.feature v11.0.2.201912141636
- openjfx.swt.feature v11.0.2.201912141636
- openjfx.web.feature v11.0.2.201912141636

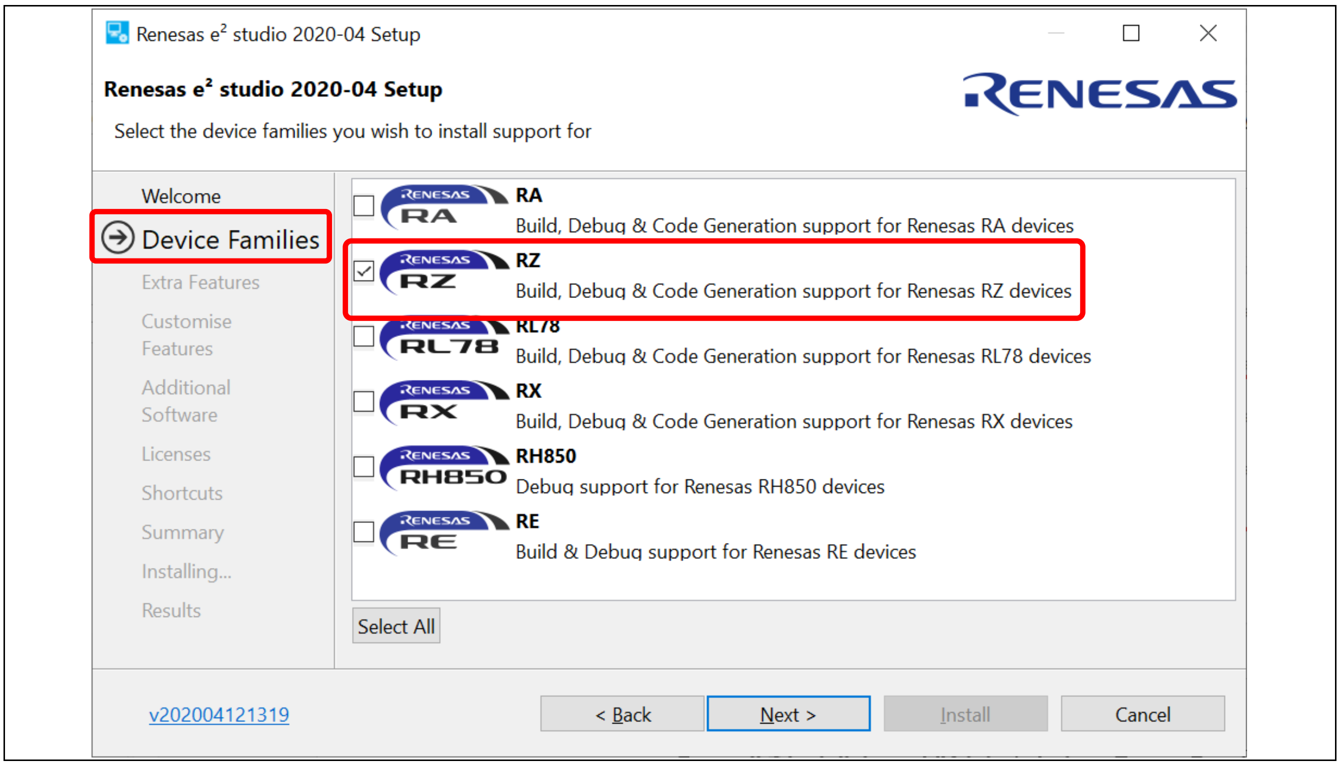
Additional components can be selected on the next page

Size of install: 446.2 MB
Temporary space required: 732.6 MB
Available: 46.9 GB

Figure 2-2 Installation of 64-bit e^2 studio – Welcome page

3. Device Families

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


Renesas e^2 studio 2020-04 Setup

Renesas e^2 studio 2020-04 Setup

Select the device families you wish to install support for

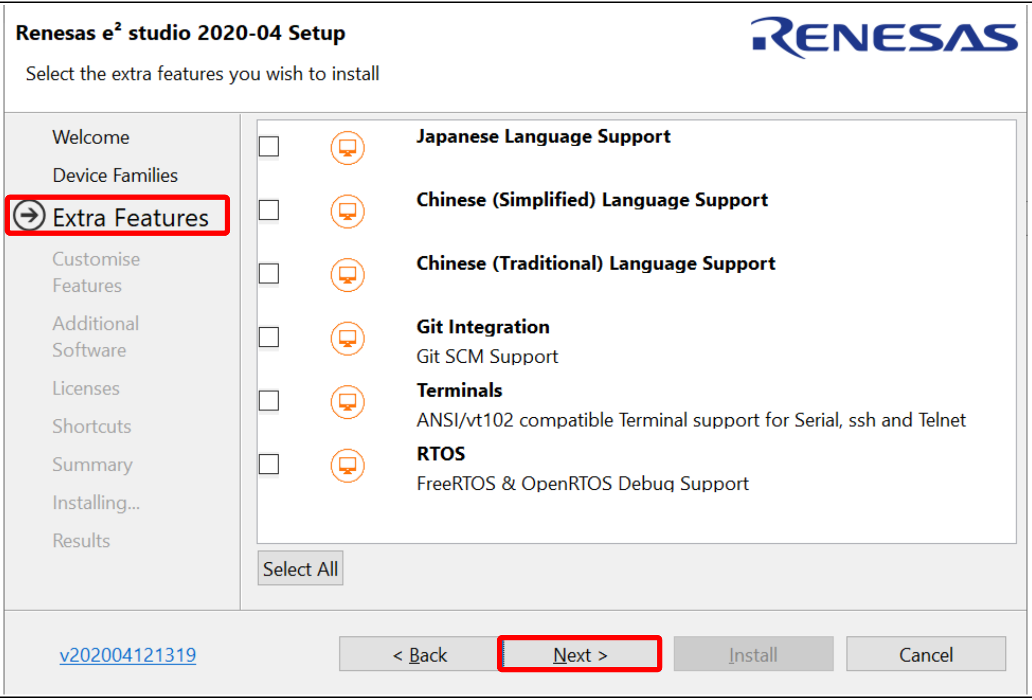
- [ ] RA – Build, Debug & Code Generation support for Renesas RA devices
- [✓] RZ – Build, Debug & Code Generation support for Renesas RZ devices
- [ ] RL78 – Build, Debug & Code Generation support for Renesas RL78 devices
- [ ] RX – Build, Debug & Code Generation support for Renesas RX devices
- [ ] RH850 – Debug support for Renesas RH850 devices
- [ ] RE – Build & Debug support for Renesas RE devices

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

4. Extra Features

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

Click the [Next] button to continue.


Renesas e^2 studio 2020-04 Setup

Select the extra features you wish to install

☐ **Japanese Language Support**

☐ **Chinese (Simplified) Language Support**

☐ **Chinese (Traditional) Language Support**

☐ **Git Integration**
Git SCM Support

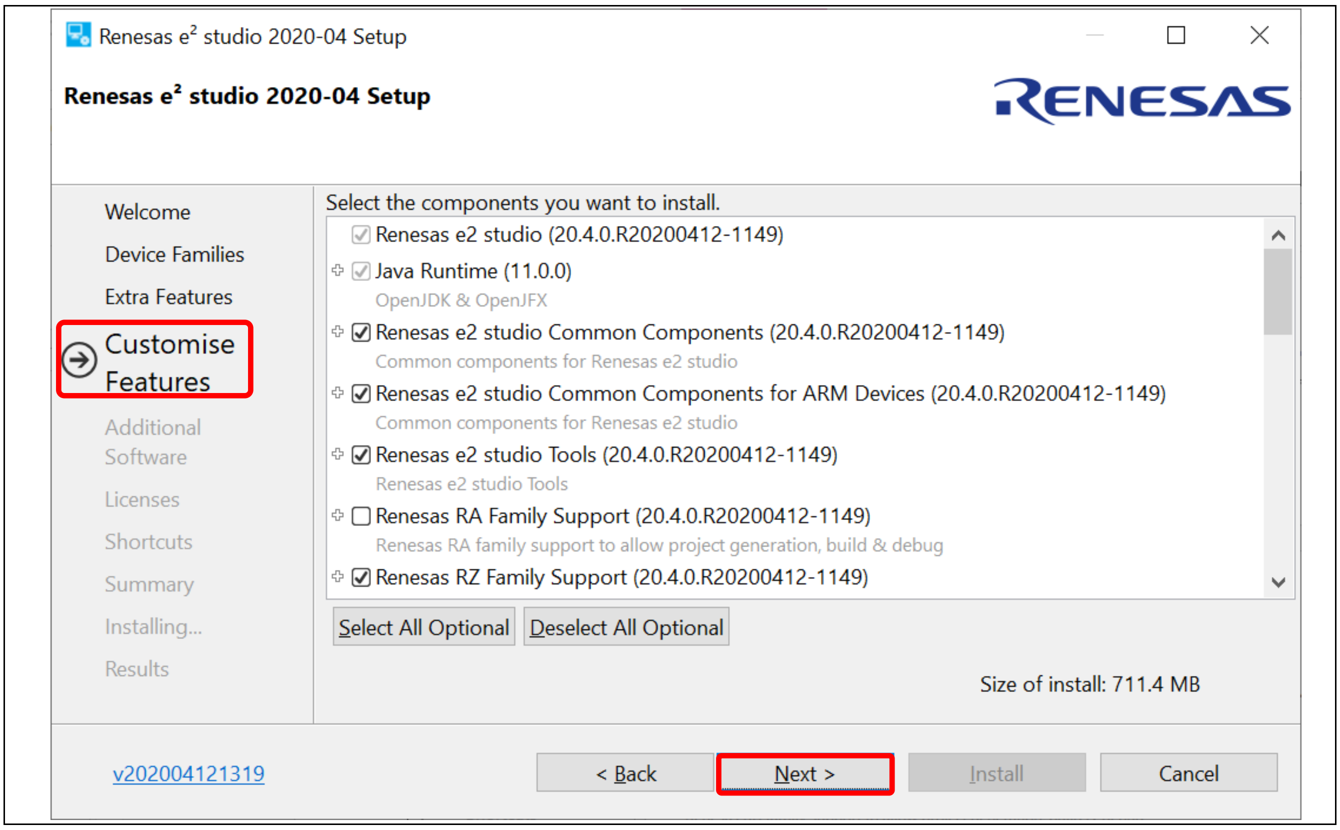
☐ **Terminals**
ANSI/vt102 compatible Terminal support for Serial, ssh and Telnet

☐ **RTOS**
FreeRTOS & OpenRTOS Debug Support

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

5. Customise Features

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


Renesas e² studio 2020-04 Setup

Renesas e² studio 2020-04 Setup

Select the components you want to install.

- ☑ Renesas e2 studio (20.4.0.R20200412-1149)
- ☑ Java Runtime (11.0.0)
  OpenJDK & OpenJFX
- ☑ Renesas e2 studio Common Components (20.4.0.R20200412-1149)
  Common components for Renesas e2 studio
- ☑ Renesas e2 studio Common Components for ARM Devices (20.4.0.R20200412-1149)
  Common components for Renesas e2 studio
- ☑ Renesas e2 studio Tools (20.4.0.R20200412-1149)
  Renesas e2 studio Tools
- ☐ Renesas RA Family Support (20.4.0.R20200412-1149)
  Renesas RA family support to allow project generation, build & debug
- ☑ Renesas RZ Family Support (20.4.0.R20200412-1149)

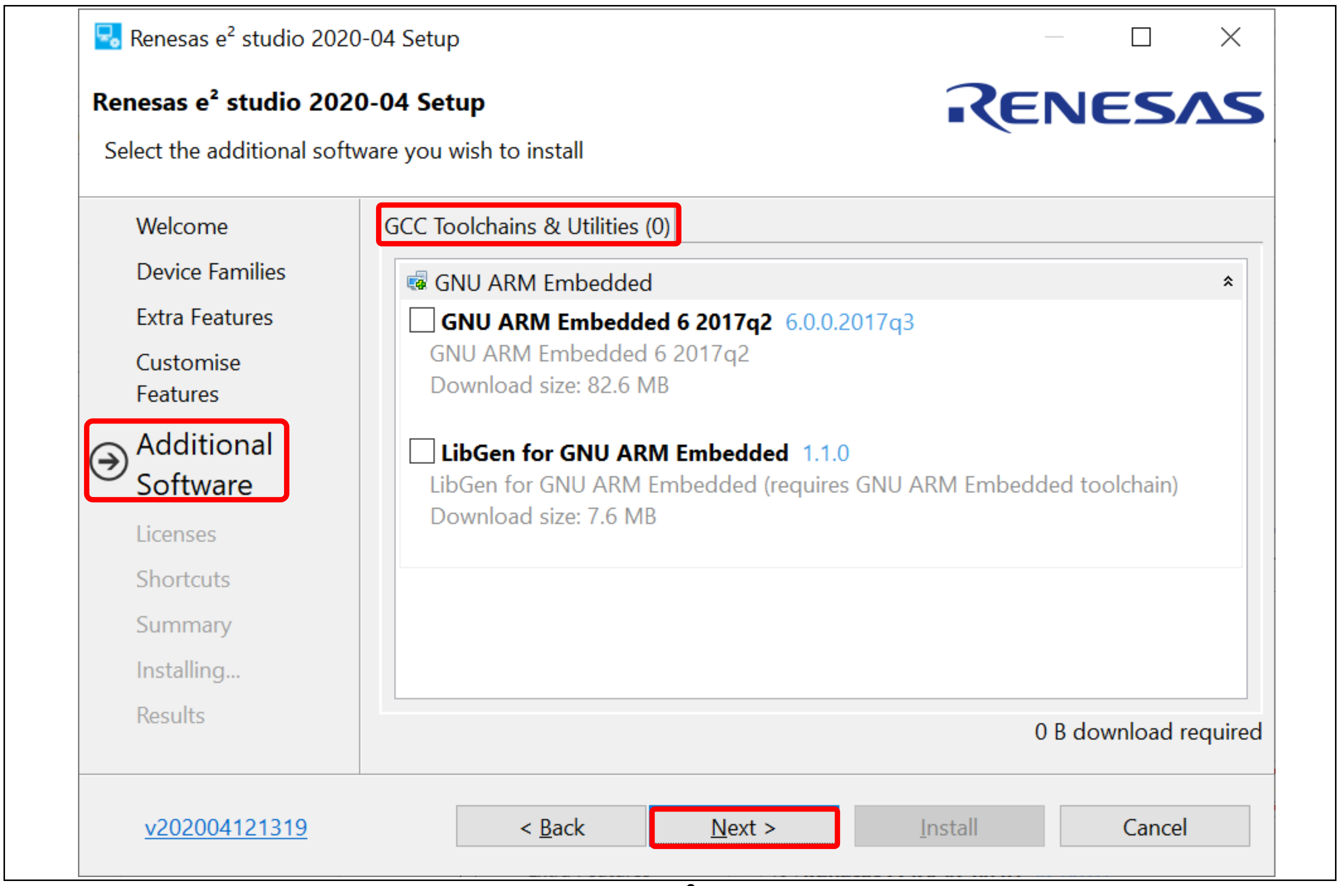
Size of install: 711.4 MB

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

6. Additional Software

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

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



Renesas e^2 studio 2020-04 Setup

Renesas e^2 studio 2020-04 Setup

Select the additional software you wish to install

GCC Toolchains & Utilities (0)

GNU ARM Embedded

☐ **GNU ARM Embedded 6 2017q2** 6.0.0.2017q3
GNU ARM Embedded 6 2017q2
Download size: 82.6 MB

☐ **LibGen for GNU ARM Embedded** 1.1.0
LibGen for GNU ARM Embedded (requires GNU ARM Embedded toolchain)
Download size: 7.6 MB

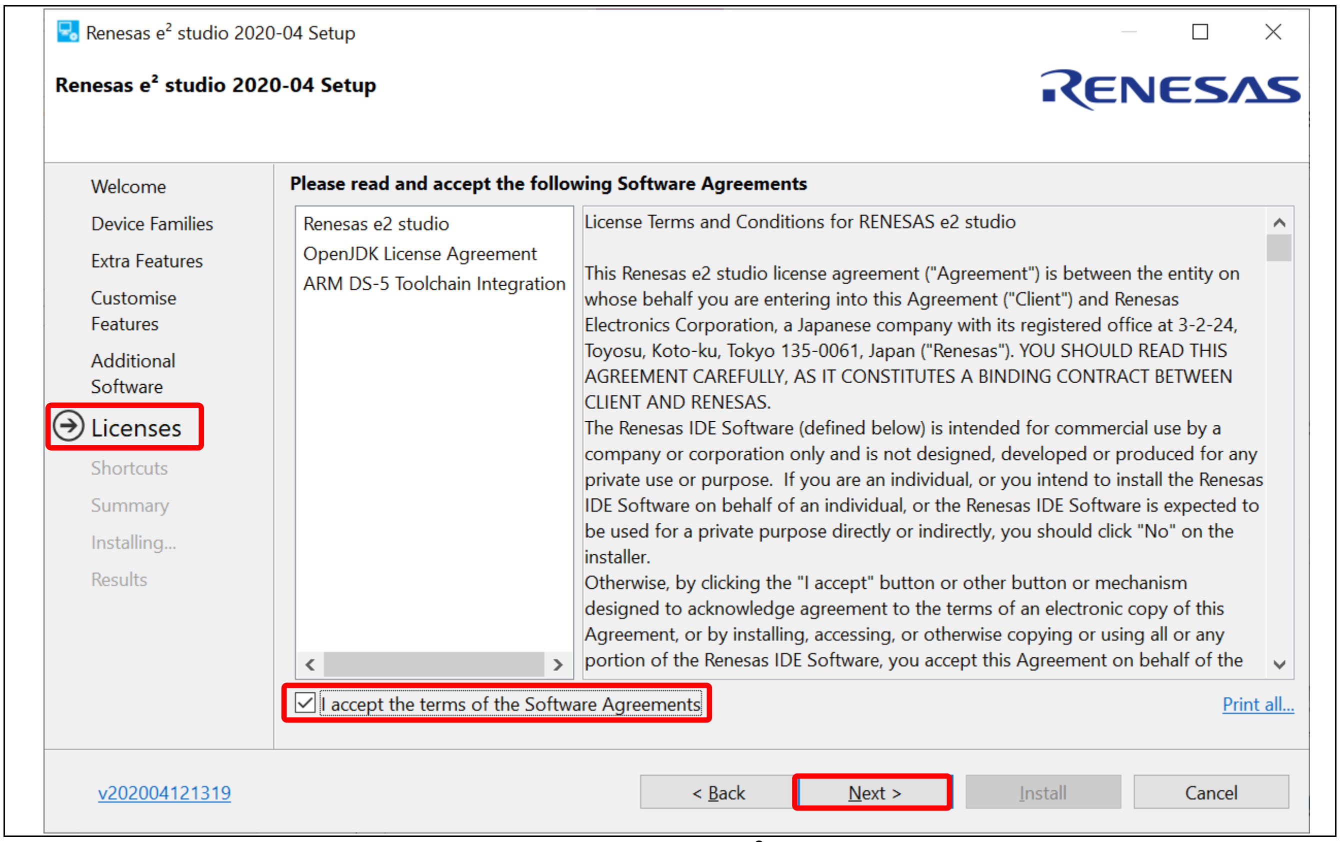
0 B download required

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

7. License Agreement

   Read and accept the software license agreement. Click the [Next] button.

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


Renesas e² studio 2020-04 Setup
Renesas e² studio 2020-04 Setup
RENESAS
Welcome
Device Families
Extra Features
Customise Features
Additional Software
Licenses
Shortcuts
Summary
Installing...
Results
Please read and accept the following Software Agreements
Renesas e2 studio
OpenJDK License Agreement
ARM DS-5 Toolchain Integration
I accept the terms of the Software Agreements
Print all...
v202004121319
< Back
Next >
Install
Cancel

License Terms and Conditions for RENESAS e2 studio

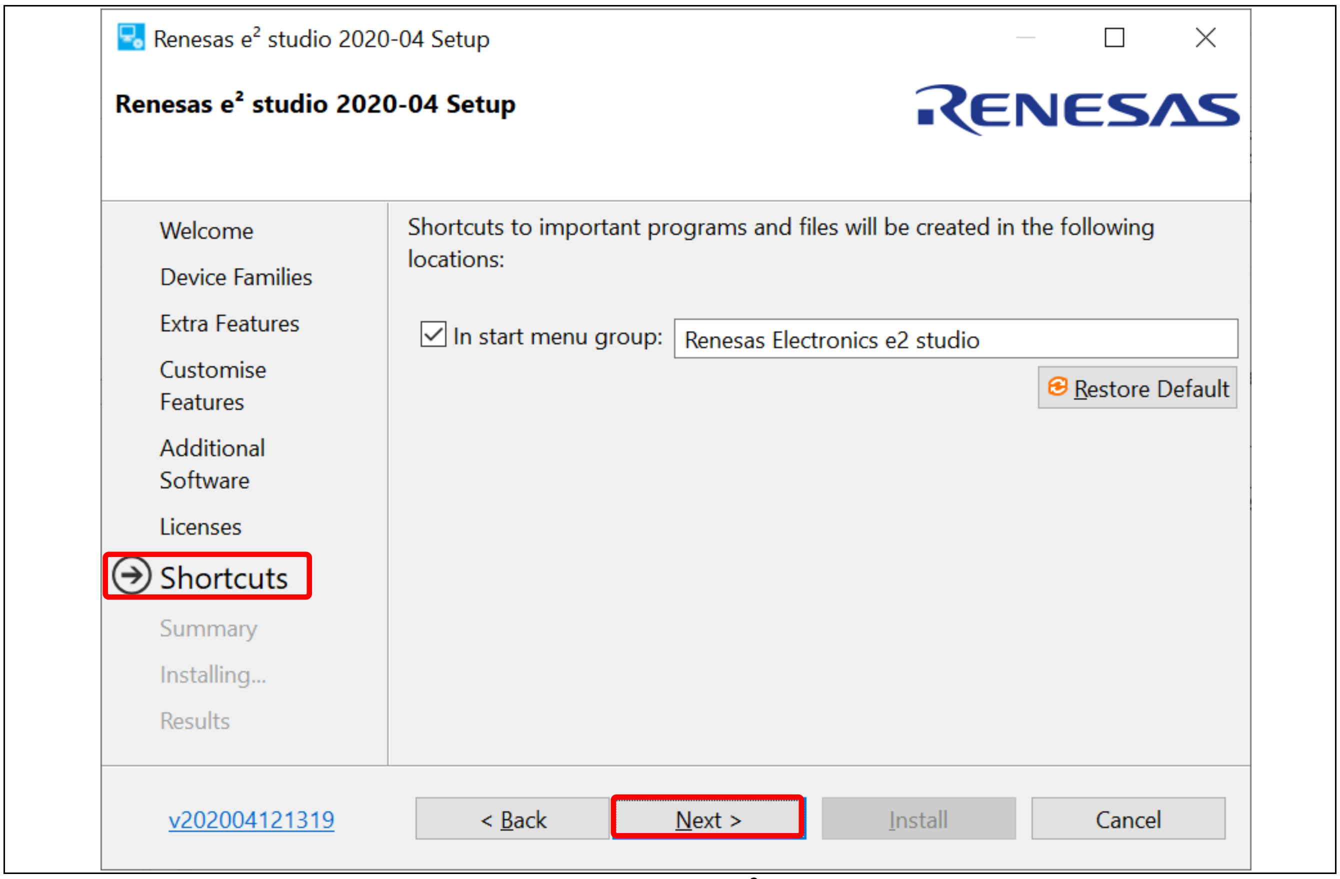
This Renesas e2 studio license agreement ("Agreement") is between the entity on whose behalf you are entering into this Agreement ("Client") and Renesas Electronics Corporation, a Japanese company with its registered office at 3-2-24, Toyosu, Koto-ku, Tokyo 135-0061, Japan ("Renesas"). YOU SHOULD READ THIS AGREEMENT CAREFULLY, AS IT CONSTITUTES A BINDING CONTRACT BETWEEN CLIENT AND RENESAS.
The Renesas IDE Software (defined below) is intended for commercial use by a company or corporation only and is not designed, developed or produced for any private use or purpose. If you are an individual, or you intend to install the Renesas IDE Software on behalf of an individual, or the Renesas IDE Software is expected to be used for a private purpose directly or indirectly, you should click "No" on the installer.
Otherwise, by clicking the "I accept" button or other button or mechanism designed to acknowledge agreement to the terms of an electronic copy of this Agreement, or by installing, accessing, or otherwise copying or using all or any portion of the Renesas IDE Software, you accept this Agreement on behalf of the

Figure 2-7 Installation of 64-bit e^2 studio – Licenses

8. Shortcuts

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

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



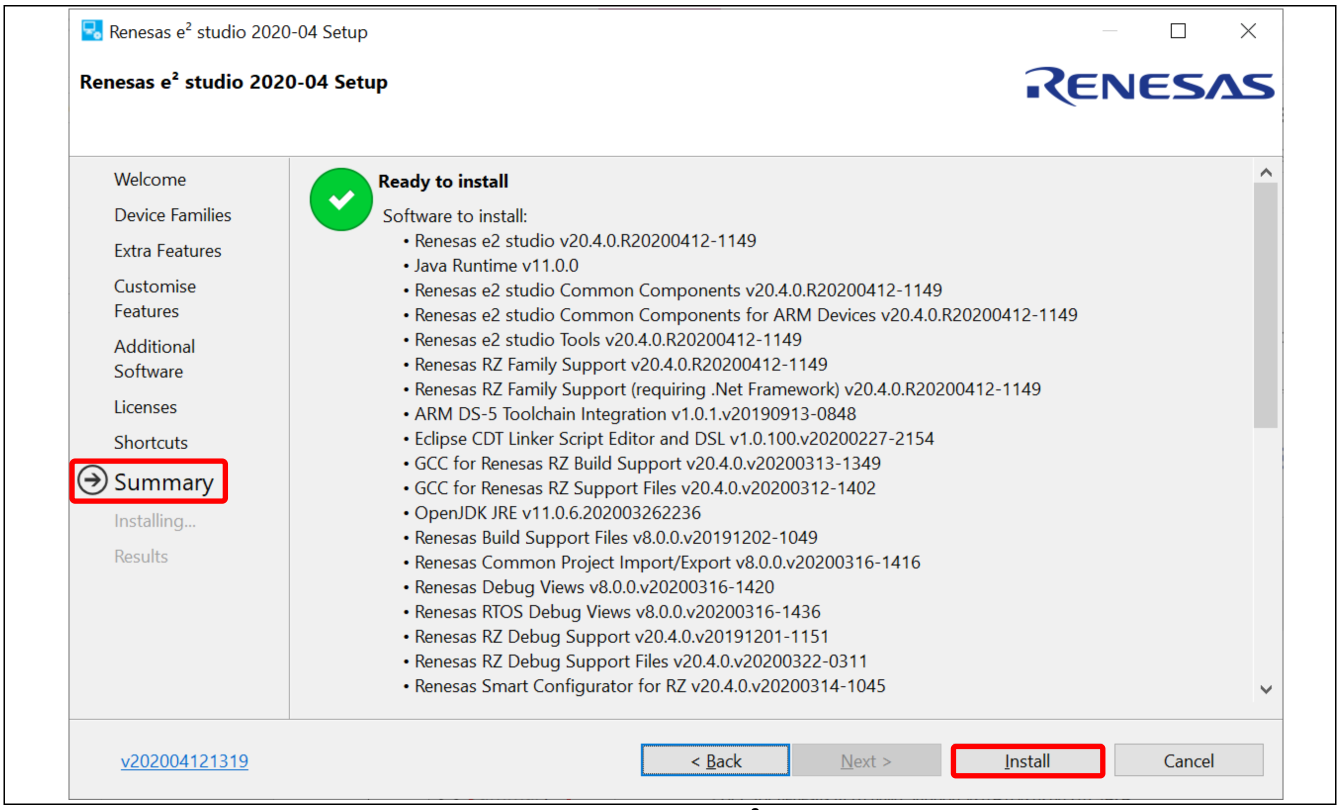
Shortcuts to important programs and files will be created in the following locations:

☑ In start menu group: Renesas Electronics e2 studio

Figure 2-8 Installation of 64-bit e^2 studio – Shortcuts

9. Summary

Components list to be installed is shown. Please confirm the contents and click the [Install] button to install the Renesas e^2 studio IDE.



Renesas e² studio 2020-04 Setup

Renesas e² studio 2020-04 Setup

Ready to install

Software to install:

- Renesas e2 studio v20.4.0.R20200412-1149
- Java Runtime v11.0.0
- Renesas e2 studio Common Components v20.4.0.R20200412-1149
- Renesas e2 studio Common Components for ARM Devices v20.4.0.R20200412-1149
- Renesas e2 studio Tools v20.4.0.R20200412-1149
- Renesas RZ Family Support v20.4.0.R20200412-1149
- Renesas RZ Family Support (requiring .Net Framework) v20.4.0.R20200412-1149
- ARM DS-5 Toolchain Integration v1.0.1.v20190913-0848
- Eclipse CDT Linker Script Editor and DSL v1.0.100.v20200227-2154
- GCC for Renesas RZ Build Support v20.4.0.v20200313-1349
- GCC for Renesas RZ Support Files v20.4.0.v20200312-1402
- OpenJDK JRE v11.0.6.202003262236
- Renesas Build Support Files v8.0.0.v20191202-1049
- Renesas Common Project Import/Export v8.0.0.v20200316-1416
- Renesas Debug Views v8.0.0.v20200316-1420
- Renesas RTOS Debug Views v8.0.0.v20200316-1436
- Renesas RZ Debug Support v20.4.0.v20191201-1151
- Renesas RZ Debug Support Files v20.4.0.v20200322-0311
- Renesas Smart Configurator for RZ v20.4.0.v20200314-1045

Figure 2-9 Installation of 64-bit e^2 studio – Summary

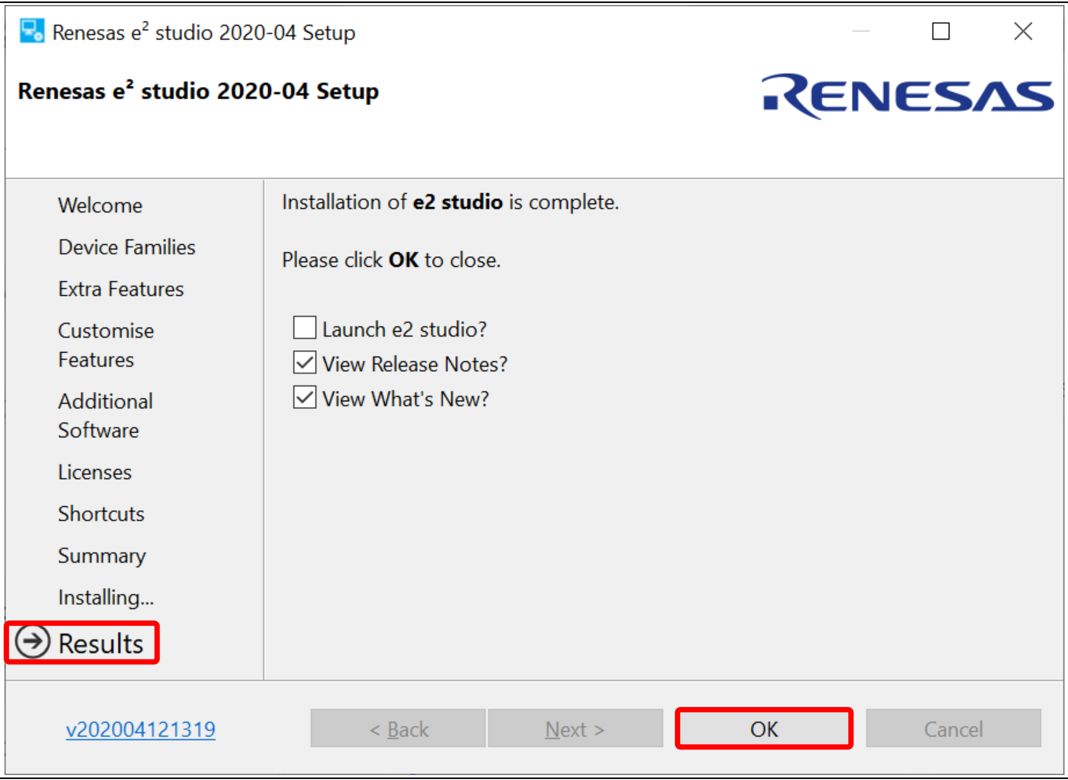
10. Installing…

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

11. Results

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

Click [OK] button to complete the installation.



Installation of **e2 studio** is complete.

Please click **OK** to close.

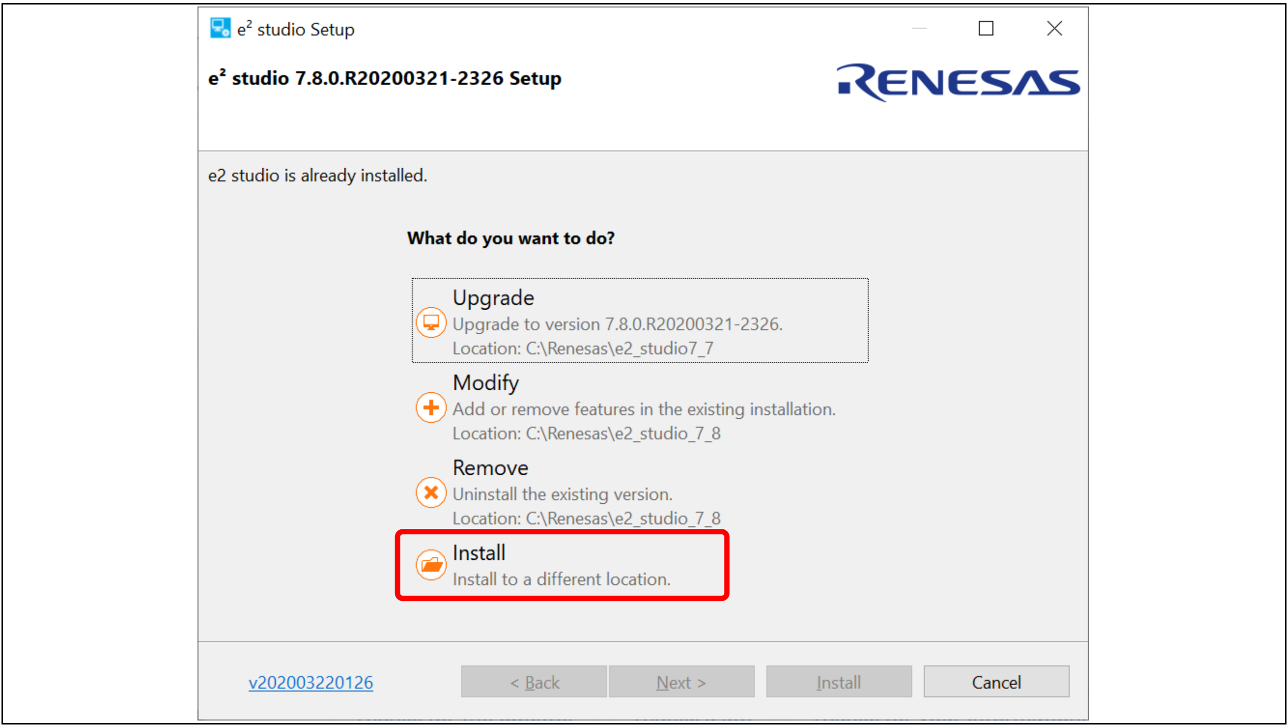
- [ ] Launch e2 studio?
- [✓] View Release Notes?
- [✓] View What's New?

Figure 2-10 Installation of 64-bit e^2 studio – Results

2.2 Installation of e^2 studio IDE (32-bit version)

1. Double-click on e^2 studio installer to invoke the e^2 studio installation wizard page. Click [Install].

   Note: If e^2 studio was installed in your PC, the option to modify, remove the existing version or install e^2 studio to a different location will be displayed. If you would like to have multiple versions of e^2 studio**, please specify "Install to a different location" here.** Click the [Next] button to continue.



e² studio Setup

e² studio 7.8.0.R20200321-2326 Setup

e2 studio is already installed.

What do you want to do?

Upgrade
Upgrade to version 7.8.0.R20200321-2326.
Location: C:\Renesas\e2_studio7_7

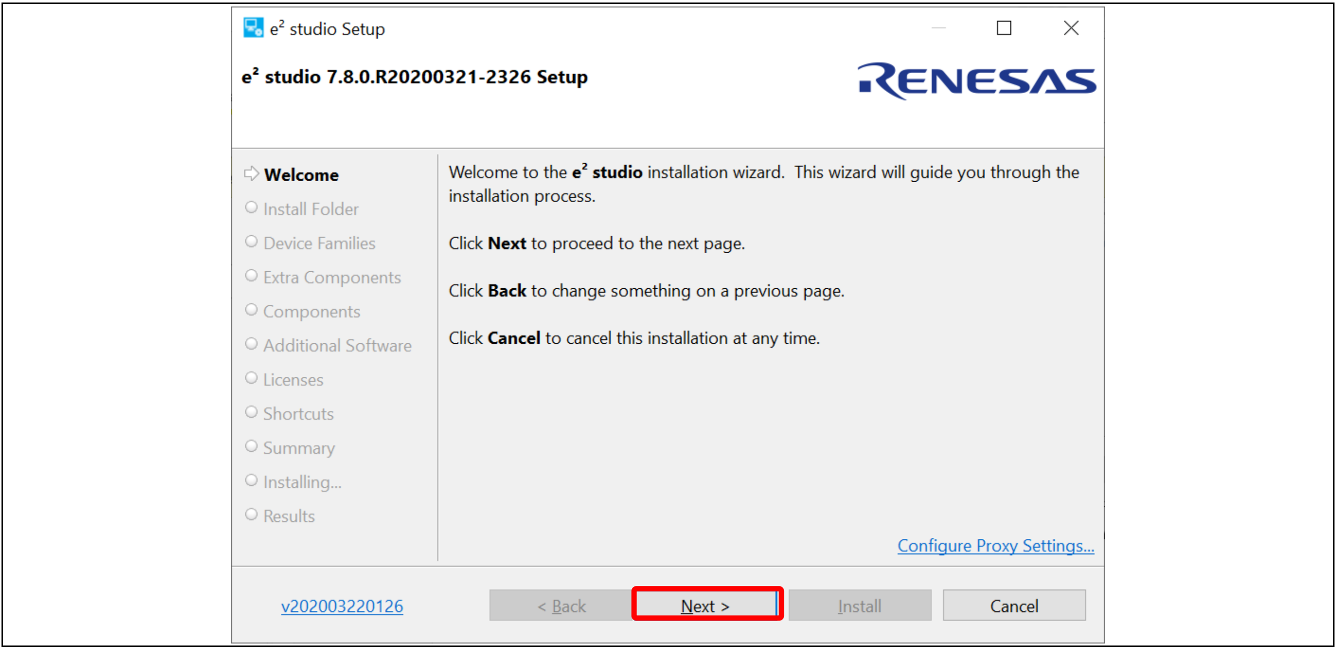
Modify
Add or remove features in the existing installation.
Location: C:\Renesas\e2_studio_7_8

Remove
Uninstall the existing version.
Location: C:\Renesas\e2_studio_7_8

Install
Install to a different location.

Figure 2-11. 32-bit e^2 studio installation wizard

2. Welcome page
   Click the [Next] button to continue.



e² studio Setup

e² studio 7.8.0.R20200321-2326 Setup

Welcome to the **e² studio** installation wizard. This wizard will guide you through the installation process.

Click **Next** to proceed to the next page.

Click **Back** to change something on a previous page.

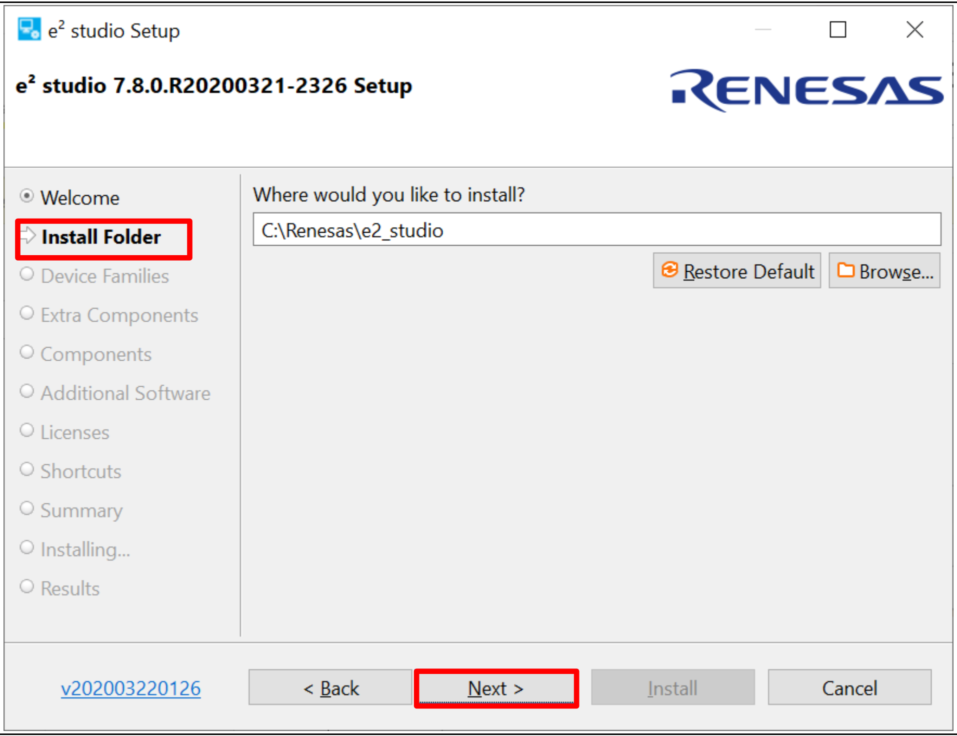
Click **Cancel** to cancel this installation at any time.

Figure 2-12. Welcome page of 32-bit installer

3. Install Folder

The default installation location is set to: `C:\Renesas\e2_studio`. Enter the install folder directly into the textbox or click the **Browse…** button to modify it. Click the **Next** button to continue.

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



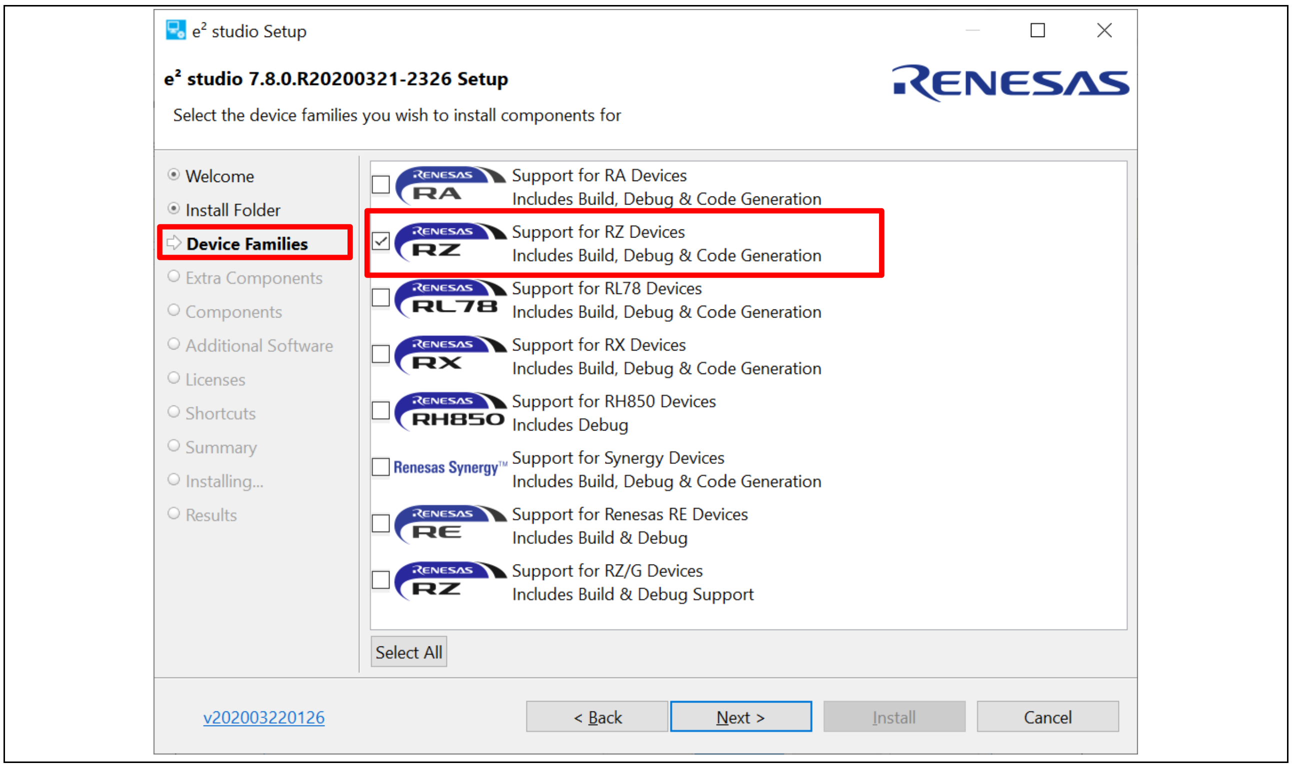
Where would you like to install?

C:\Renesas\e2_studio

Figure 2-13. Installation of 32-bit e^2 studio – Install Folder

4. Device Families

Select Devices Families to install. Make sure you tick the checkbox for **Support for RZ Devices**.


e² studio Setup

e² studio 7.8.0.R20200321-2326 Setup

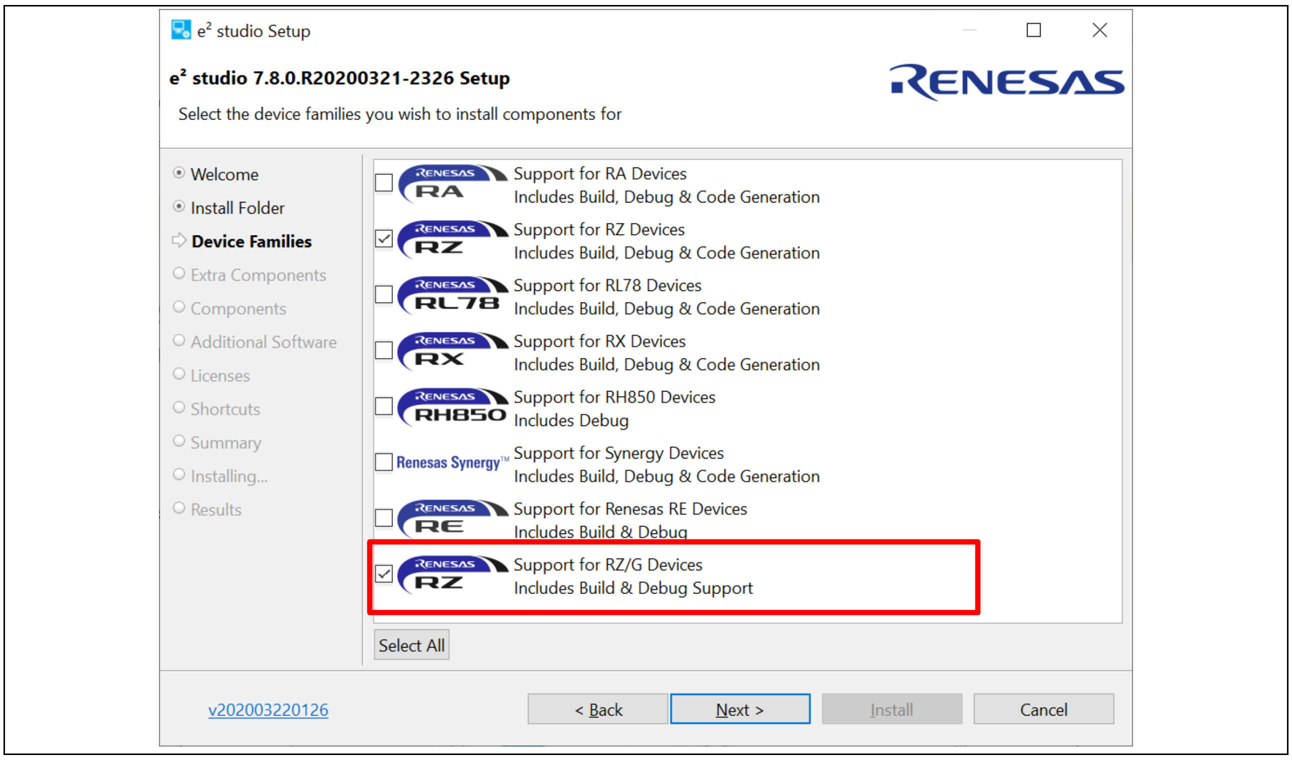
Select the device families you wish to install components for

- Welcome
- Install Folder
- **Device Families**
- Extra Components
- Components
- Additional Software
- Licenses
- Shortcuts
- Summary
- Installing...
- Results

- [ ] Support for RA Devices — Includes Build, Debug & Code Generation
- [✓] Support for RZ Devices — Includes Build, Debug & Code Generation
- [ ] Support for RL78 Devices — Includes Build, Debug & Code Generation
- [ ] Support for RX Devices — Includes Build, Debug & Code Generation
- [ ] Support for RH850 Devices — Includes Debug
- [ ] Support for Synergy Devices — Includes Build, Debug & Code Generation
- [ ] Support for Renesas RE Devices — Includes Build & Debug
- [ ] Support for RZ/G Devices — Includes Build & Debug Support

Figure 2-14. Device Families Page (32-bit): Support for RZ Devices

Note: *Support for RZ/G Devices* is the option for RZ/G Linux application project generation and debugging. (Only 32-bit version of e^2 studio has this option.) If you need to develop applications for Linux on RZ/G device, please select this too.



e^2 studio Setup

e^2 studio 7.8.0.R20200321-2326 Setup

Select the device families you wish to install components for

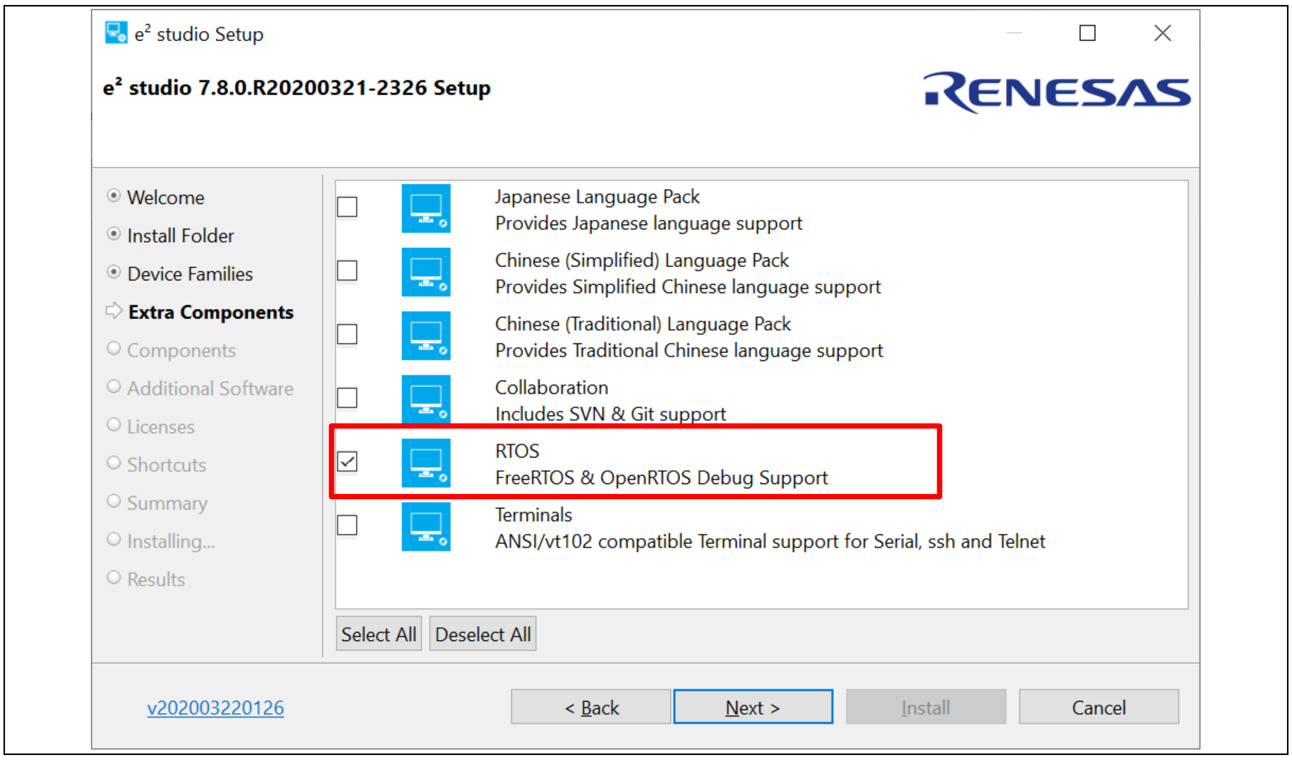
- Welcome
- Install Folder
- **Device Families**
- Extra Components
- Components
- Additional Software
- Licenses
- Shortcuts
- Summary
- Installing...
- Results

☐ RA	Support for RA Devices Includes Build, Debug & Code Generation
☑ RZ	Support for RZ Devices Includes Build, Debug & Code Generation
☐ RL78	Support for RL78 Devices Includes Build, Debug & Code Generation
☐ RX	Support for RX Devices Includes Build, Debug & Code Generation
☐ RH850	Support for RH850 Devices Includes Debug
☐ Renesas Synergy™	Support for Synergy Devices Includes Build, Debug & Code Generation
☐ RE	Support for Renesas RE Devices Includes Build & Debug
☑ RZ	Support for RZ/G Devices Includes Build & Debug Support

Figure 2-15. Device Family Page (32-bit): Support for RZ/G Devices

5. Extra Components

Select the required extra components (e.g. language packs, SVN & Git support, RTOS support…).
For example, select **RTOS** if using a RTOS such as FreeRTOS or OpenRTOS.
Click the **Next** button to continue.


e^2 studio Setup

e^2 studio 7.8.0.R20200321-2326 Setup

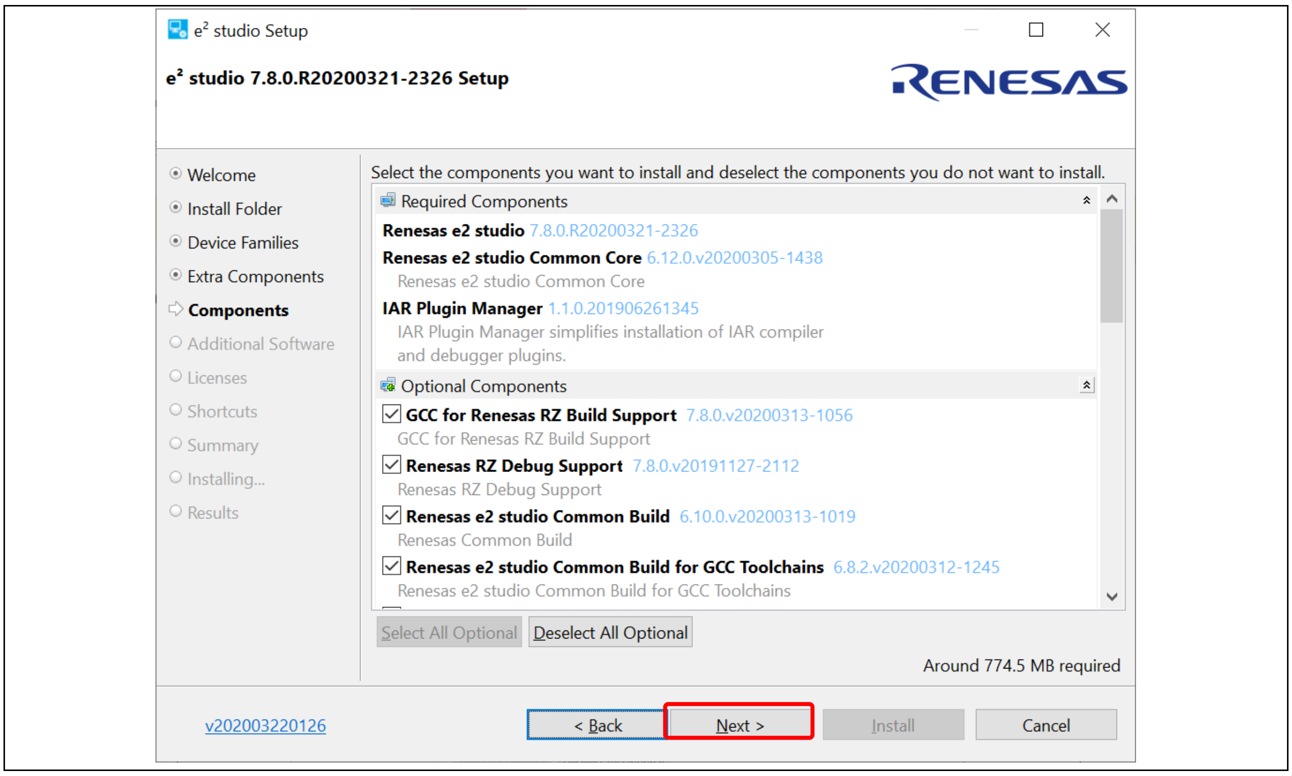
- Welcome
- Install Folder
- Device Families
- **Extra Components**
- Components
- Additional Software
- Licenses
- Shortcuts
- Summary
- Installing...
- Results

- [ ] Japanese Language Pack
  Provides Japanese language support
- [ ] Chinese (Simplified) Language Pack
  Provides Simplified Chinese language support
- [ ] Chinese (Traditional) Language Pack
  Provides Traditional Chinese language support
- [ ] Collaboration
  Includes SVN & Git support
- [✓] RTOS
  FreeRTOS & OpenRTOS Debug Support
- [ ] Terminals
  ANSI/vt102 compatible Terminal support for Serial, ssh and Telnet

Figure 2-16. Extra Components Page (32-bit): RTOS is selected

6. Components

The components needed by the selected **Device Families**, are automatically selected.
Please check the selected components and then click the **Next** button to continue.


e² studio Setup

e² studio 7.8.0.R20200321-2326 Setup

Select the components you want to install and deselect the components you do not want to install.

Required Components

Renesas e2 studio 7.8.0.R20200321-2326

Renesas e2 studio Common Core 6.12.0.v20200305-1438
Renesas e2 studio Common Core

IAR Plugin Manager 1.1.0.201906261345
IAR Plugin Manager simplifies installation of IAR compiler and debugger plugins.

Optional Components

☑ **GCC for Renesas RZ Build Support** 7.8.0.v20200313-1056
GCC for Renesas RZ Build Support

☑ **Renesas RZ Debug Support** 7.8.0.v20191127-2112
Renesas RZ Debug Support

☑ **Renesas e2 studio Common Build** 6.10.0.v20200313-1019
Renesas Common Build

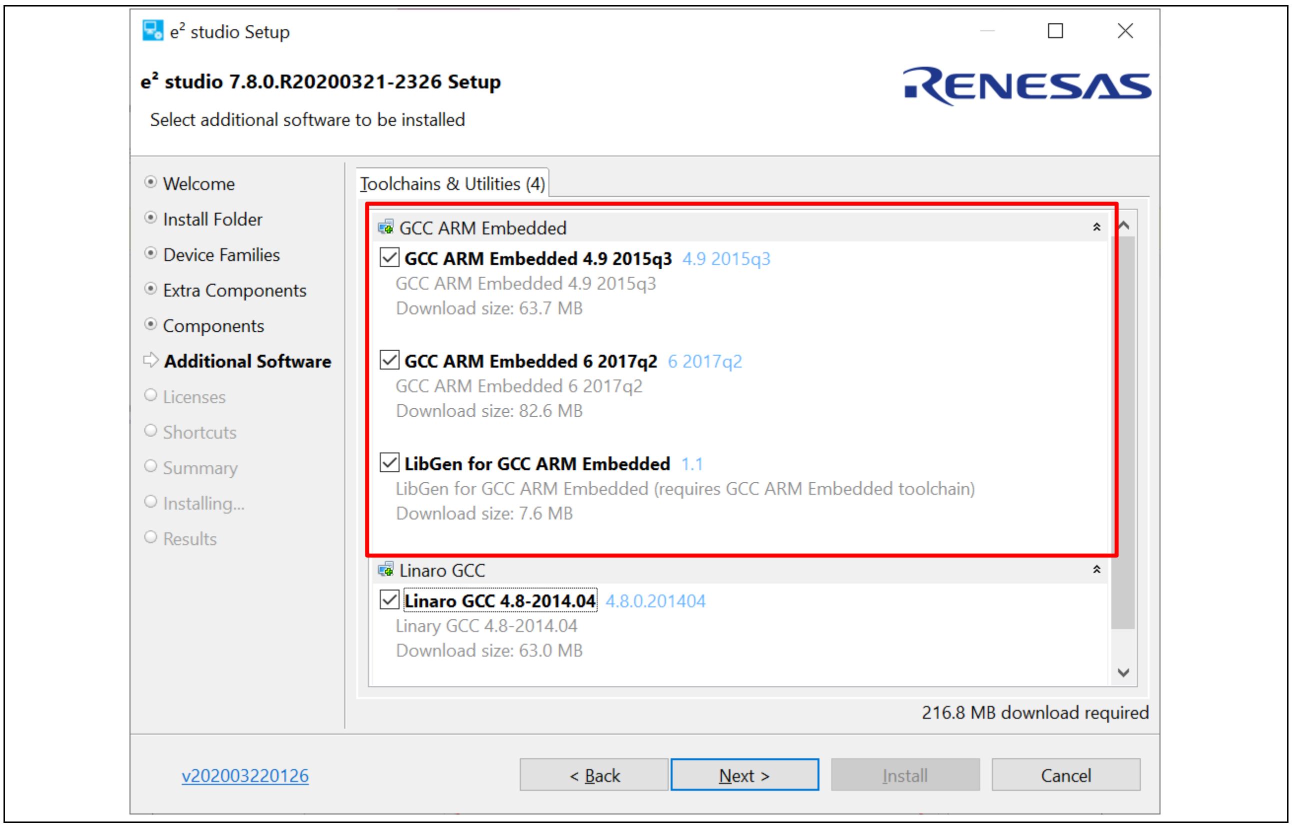
☑ **Renesas e2 studio Common Build for GCC Toolchains** 6.8.2.v20200312-1245
Renesas e2 studio Common Build for GCC Toolchains

Around 774.5 MB required

v202003220126

Figure 2-17. Components Page (32-bit e² studio)

7. Additional Software
   Check that additional software (i.e. GCC ARM Embedded (it is same with GNU ARM Embedded toolchain) and LibGen for GCC ARM Embedded) are selected. The additional software list is automatically populated according to the selected device families. The selected software's installer will be invoked by the e^2 studio installer (for more information, see chapter 2.3). If you want to use a version of GCC ARM Embedded which is not included in the list, please download and install it separately (see chapter 1.3). Click the **Next** button to continue.


e^2 studio Setup

e^2 studio 7.8.0.R20200321-2326 Setup

Select additional software to be installed

- Welcome
- Install Folder
- Device Families
- Extra Components
- Components
- **Additional Software**
- Licenses
- Shortcuts
- Summary
- Installing...
- Results

Toolchains & Utilities (4)

GCC ARM Embedded

☑ **GCC ARM Embedded 4.9 2015q3** 4.9 2015q3
GCC ARM Embedded 4.9 2015q3
Download size: 63.7 MB

☑ **GCC ARM Embedded 6 2017q2** 6 2017q2
GCC ARM Embedded 6 2017q2
Download size: 82.6 MB

☑ **LibGen for GCC ARM Embedded** 1.1
LibGen for GCC ARM Embedded (requires GCC ARM Embedded toolchain)
Download size: 7.6 MB

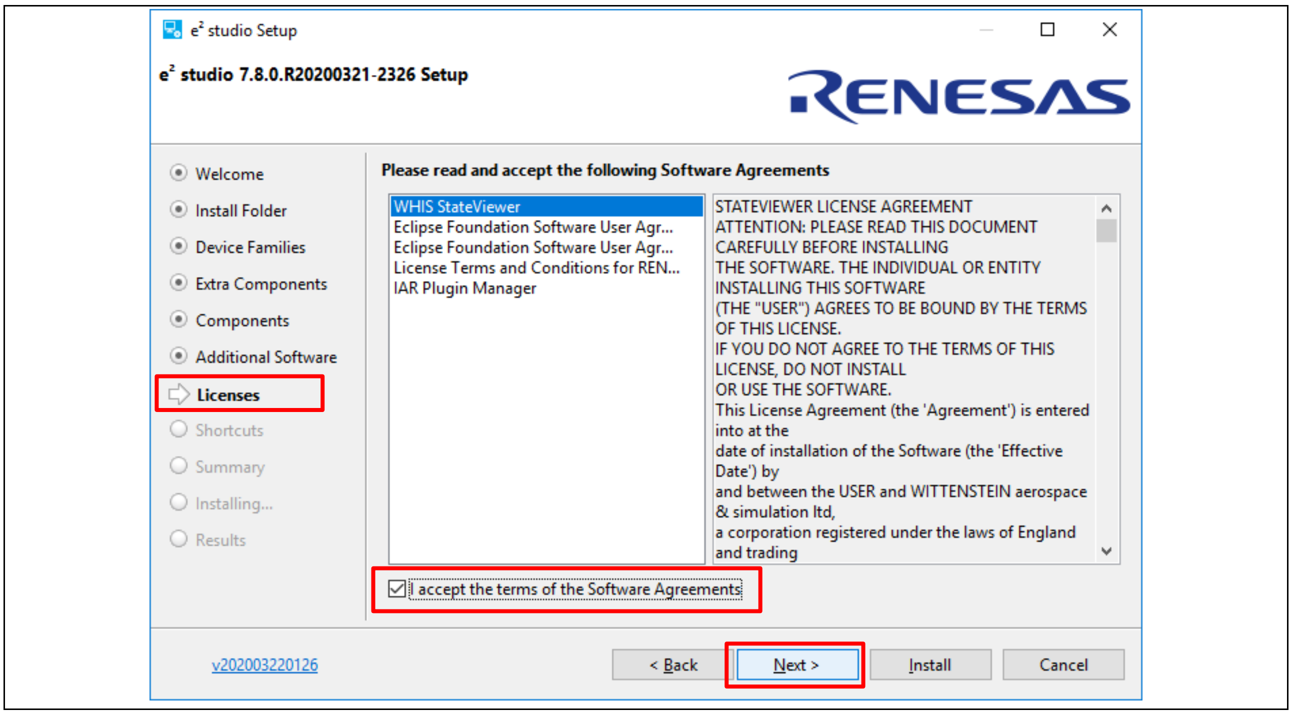
Linaro GCC

☑ **Linaro GCC 4.8-2014.04** 4.8.0.201404
Linary GCC 4.8-2014.04
Download size: 63.0 MB

216.8 MB download required

Figure 2-18. Additional Software Page (32-bit e^2 studio)

8. License Agreement
   Read and accept the software license agreement by selecting the **Next** button.
   Please note that you have to accept the license agreement, otherwise installation cannot be continued.



Please read and accept the following Software Agreements

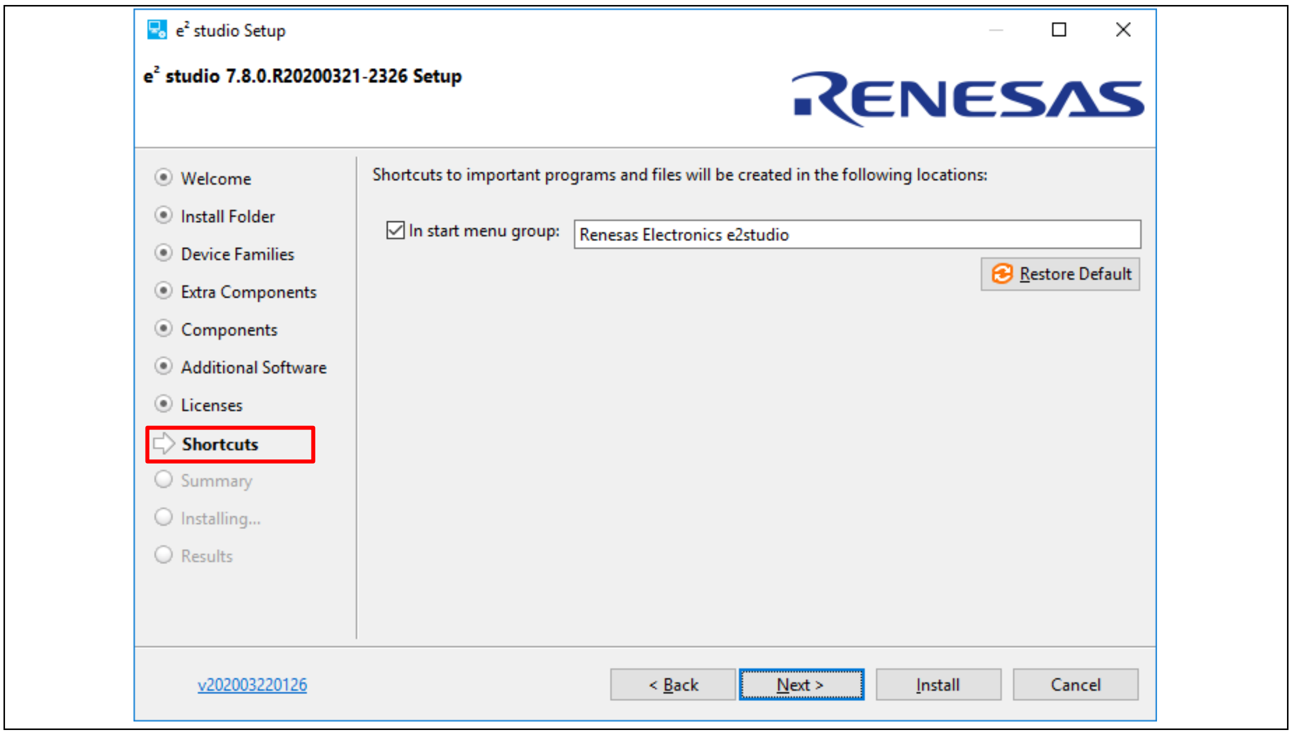
WHIS StateViewer
Eclipse Foundation Software User Agr...
Eclipse Foundation Software User Agr...
License Terms and Conditions for REN...
IAR Plugin Manager

STATEVIEWER LICENSE AGREEMENT
ATTENTION: PLEASE READ THIS DOCUMENT
CAREFULLY BEFORE INSTALLING
THE SOFTWARE. THE INDIVIDUAL OR ENTITY
INSTALLING THIS SOFTWARE
(THE "USER") AGREES TO BE BOUND BY THE TERMS
OF THIS LICENSE.
IF YOU DO NOT AGREE TO THE TERMS OF THIS
LICENSE, DO NOT INSTALL
OR USE THE SOFTWARE.
This License Agreement (the 'Agreement') is entered
into at the
date of installation of the Software (the 'Effective
Date') by
and between the USER and WITTENSTEIN aerospace
& simulation ltd,
a corporation registered under the laws of England
and trading

☑ I accept the terms of the Software Agreements

Figure 2-19. Additional Software Page (32-bit e^2 studio)

9. Shortcuts
   Select a shortcut name for the start menu and click the **Next** button to continue.

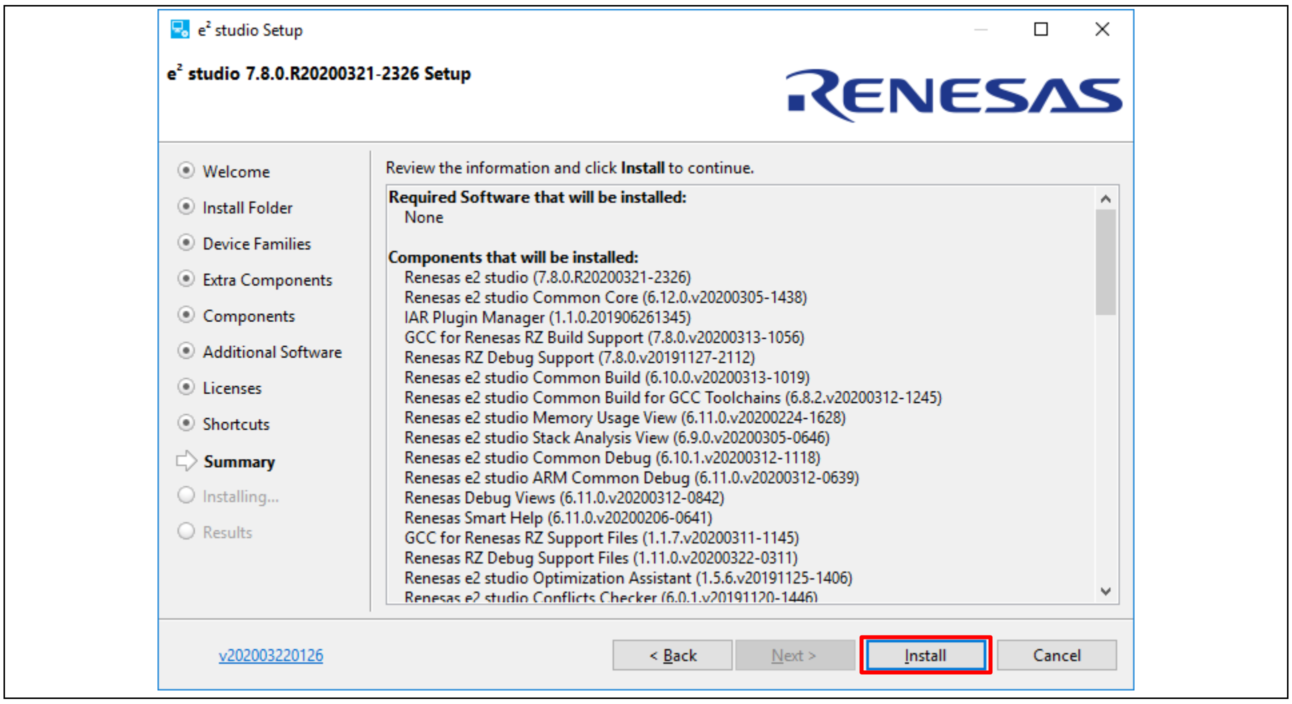


Shortcuts to important programs and files will be created in the following locations:

☑ In start menu group: Renesas Electronics e2studio

Figure 2-20. Shortcuts Page (32-bit e^2 studio)

10. Summary
    Click the **Install** button to install the Renesas e^2 studio IDE.



e^2 studio 7.8.0.R20200321-2326 Setup

Review the information and click **Install** to continue.

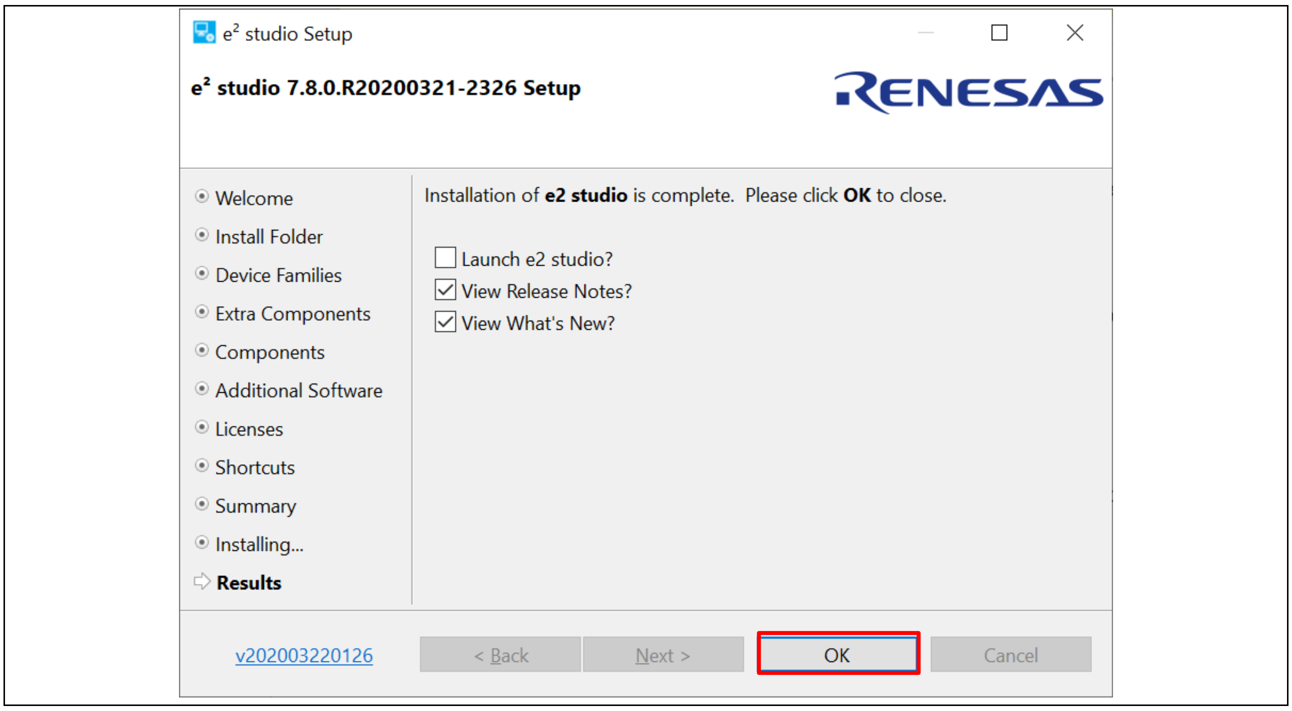
Required Software that will be installed:
None

Components that will be installed:
Renesas e2 studio (7.8.0.R20200321-2326)
Renesas e2 studio Common Core (6.12.0.v20200305-1438)
IAR Plugin Manager (1.1.0.201906261345)
GCC for Renesas RZ Build Support (7.8.0.v20200313-1056)
Renesas RZ Debug Support (7.8.0.v20191127-2112)
Renesas e2 studio Common Build (6.10.0.v20200313-1019)
Renesas e2 studio Common Build for GCC Toolchains (6.8.2.v20200312-1245)
Renesas e2 studio Memory Usage View (6.11.0.v20200224-1628)
Renesas e2 studio Stack Analysis View (6.9.0.v20200305-0646)
Renesas e2 studio Common Debug (6.10.1.v20200312-1118)
Renesas e2 studio ARM Common Debug (6.11.0.v20200312-0639)
Renesas Debug Views (6.11.0.v20200312-0842)
Renesas Smart Help (6.11.0.v20200206-0641)
GCC for Renesas RZ Support Files (1.1.7.v20200311-1145)
Renesas RZ Debug Support Files (1.11.0.v20200322-0311)
Renesas e2 studio Optimization Assistant (1.5.6.v20191125-1406)
Renesas e2 studio Conflicts Checker (6.0.1.v20191120-1446)

Figure 2-21. Summary Page (32-bit e^2 studio)

11. Installing…
    The installation is performed. Depending on selected items of additional software, new dialog prompts may appear during the installation process. Please see chapter 2.2 for more detailed information.

12. Results
    Click the **OK** button to complete the installation.



e^2 studio 7.8.0.R20200321-2326 Setup

Installation of **e2 studio** is complete. Please click **OK** to close.

☐ Launch e2 studio?
☑ View Release Notes?
☑ View What's New?

Figure 2-22. Summary Page (32-bit e^2 studio)

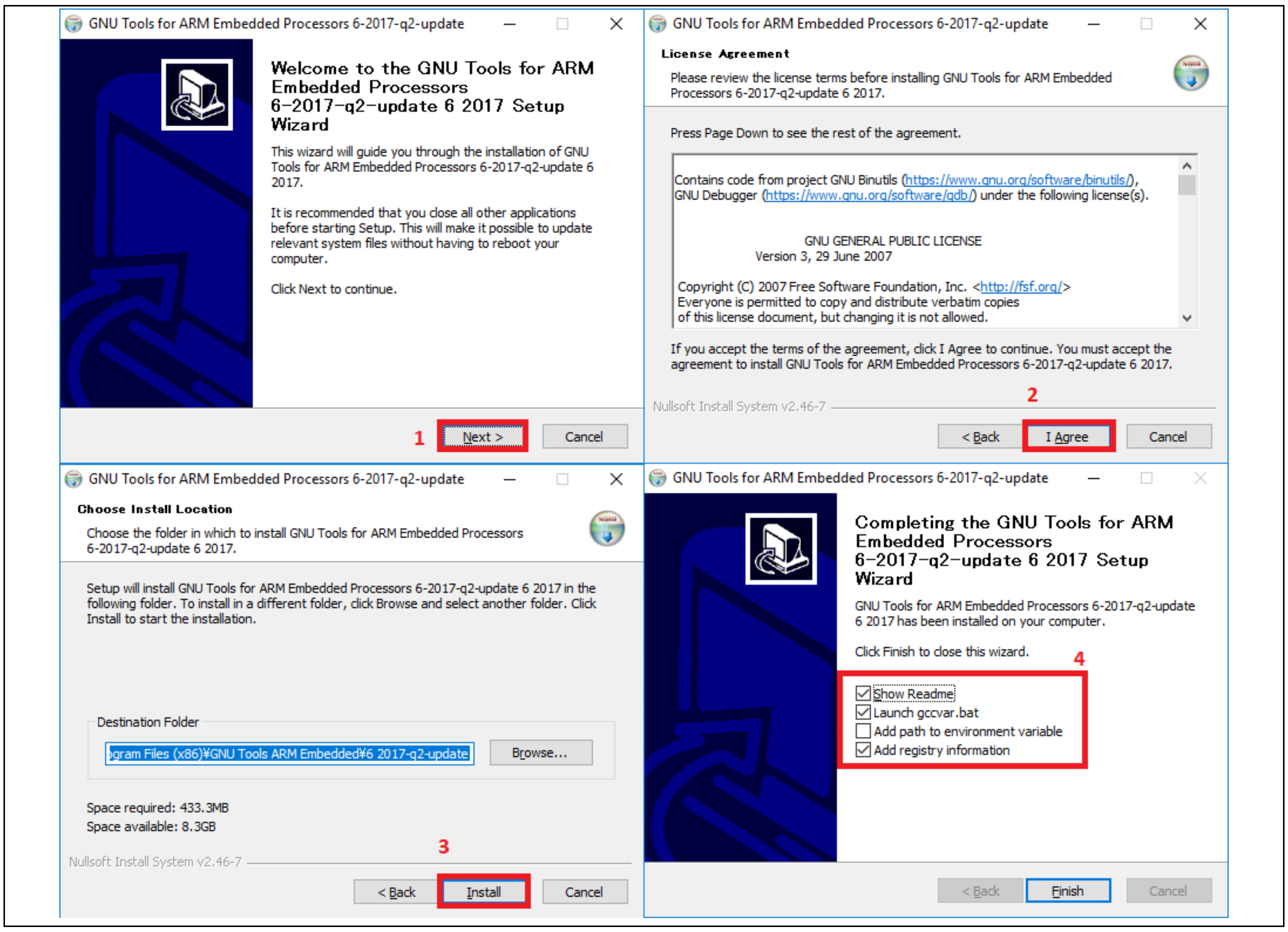
2.3 Installation of Additional Software

The e^2 studio installer invokes other installers which were selected within the **Additional Software** pane (such as GNU Compilers or **Libgen Update for GNU ARM Embedded Toolchains**).

2.3.1 GNU ARM Embedded Toolchain

If it was selected in the **Additional Software** pane of e^2 studio, you will see the installation wizard for the GNU ARM Embedded Toolchain during the installation process.

1. Click **Next**.
2. After reading the license agreement, click **I Agree**.
3. Select the destination folder. The default folder is recommended.
4. At the end of the installation, you can select options such as adding the path to the Windows environment variable, and adding registry information.



Welcome to the GNU Tools for ARM Embedded Processors 6-2017-q2-update 6 2017 Setup Wizard

This wizard will guide you through the installation of GNU Tools for ARM Embedded Processors 6-2017-q2-update 6 2017.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

License Agreement

Please review the license terms before installing GNU Tools for ARM Embedded Processors 6-2017-q2-update 6 2017.

Press Page Down to see the rest of the agreement.

Contains code from project GNU Binutils (https://www.gnu.org/software/binutils/), GNU Debugger (https://www.gnu.org/software/gdb/) under the following license(s).

GNU GENERAL PUBLIC LICENSE
Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install GNU Tools for ARM Embedded Processors 6-2017-q2-update 6 2017.

Nullsoft Install System v2.46-7

Choose Install Location

Choose the folder in which to install GNU Tools for ARM Embedded Processors 6-2017-q2-update 6 2017.

Setup will install GNU Tools for ARM Embedded Processors 6-2017-q2-update 6 2017 in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

Destination Folder
gram Files (x86)¥GNU Tools ARM Embedded¥6 2017-q2-update

Space required: 433.3MB
Space available: 8.3GB

Nullsoft Install System v2.46-7

Completing the GNU Tools for ARM Embedded Processors 6-2017-q2-update 6 2017 Setup Wizard

GNU Tools for ARM Embedded Processors 6-2017-q2-update 6 2017 has been installed on your computer.

Click Finish to close this wizard.

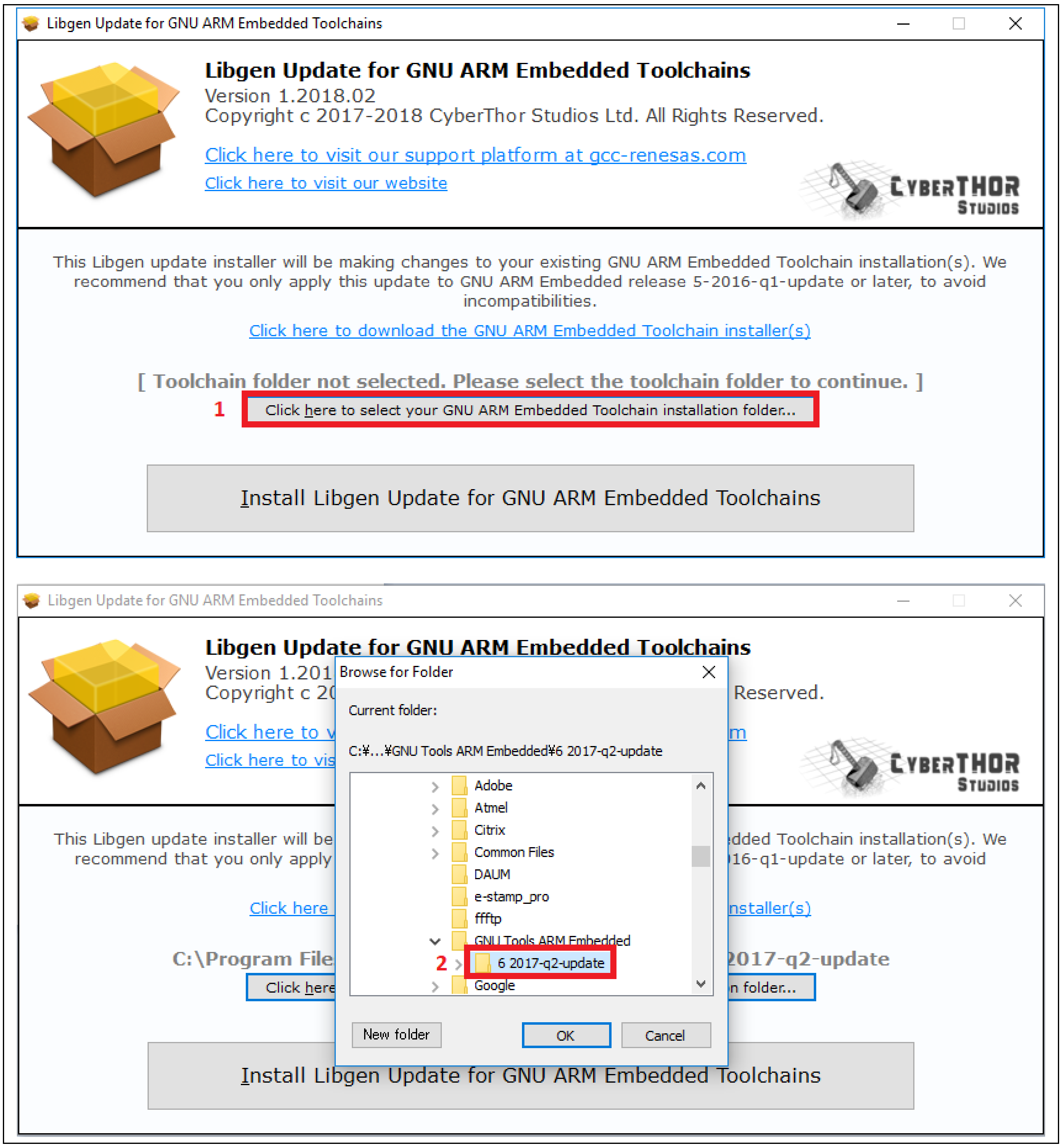
- ☑ Show Readme
- ☑ Launch gccvar.bat
- ☐ Add path to environment variable
- ☑ Add registry information

Figure 2-23. Installation of GNU ARM Embedded Toolchain

2.3.2 Libgen Update for GNU ARM Embedded Toolchains

If it was selected in the **Additional Software** pane of e^2 studio, you will see the installation wizard for the **Libgen Update for GNU ARM Embedded Toolchains** during the installation process. If you do not wish to update the GNU ARM Embedded Toolchain, terminate the **Libgen Update for GNU ARM Embedded Toolchains** by clicking the ✕ (close) button.

1. Click **Click here to select your GNU ARM Embedded Toolchain installation folder**.
2. Select the folder for the **GNU ARM Embedded Toolchain** to update.


Libgen Update for GNU ARM Embedded Toolchains

Libgen Update for GNU ARM Embedded Toolchains
Version 1.2018.02
Copyright c 2017-2018 CyberThor Studios Ltd. All Rights Reserved.
Click here to visit our support platform at gcc-renesas.com
Click here to visit our website

This Libgen update installer will be making changes to your existing GNU ARM Embedded Toolchain installation(s). We recommend that you only apply this update to GNU ARM Embedded release 5-2016-q1-update or later, to avoid incompatibilities.

Click here to download the GNU ARM Embedded Toolchain installer(s)

[Toolchain folder not selected. Please select the toolchain folder to continue.]

1 Click here to select your GNU ARM Embedded Toolchain installation folder...

Install Libgen Update for GNU ARM Embedded Toolchains

Browse for Folder

Current folder:

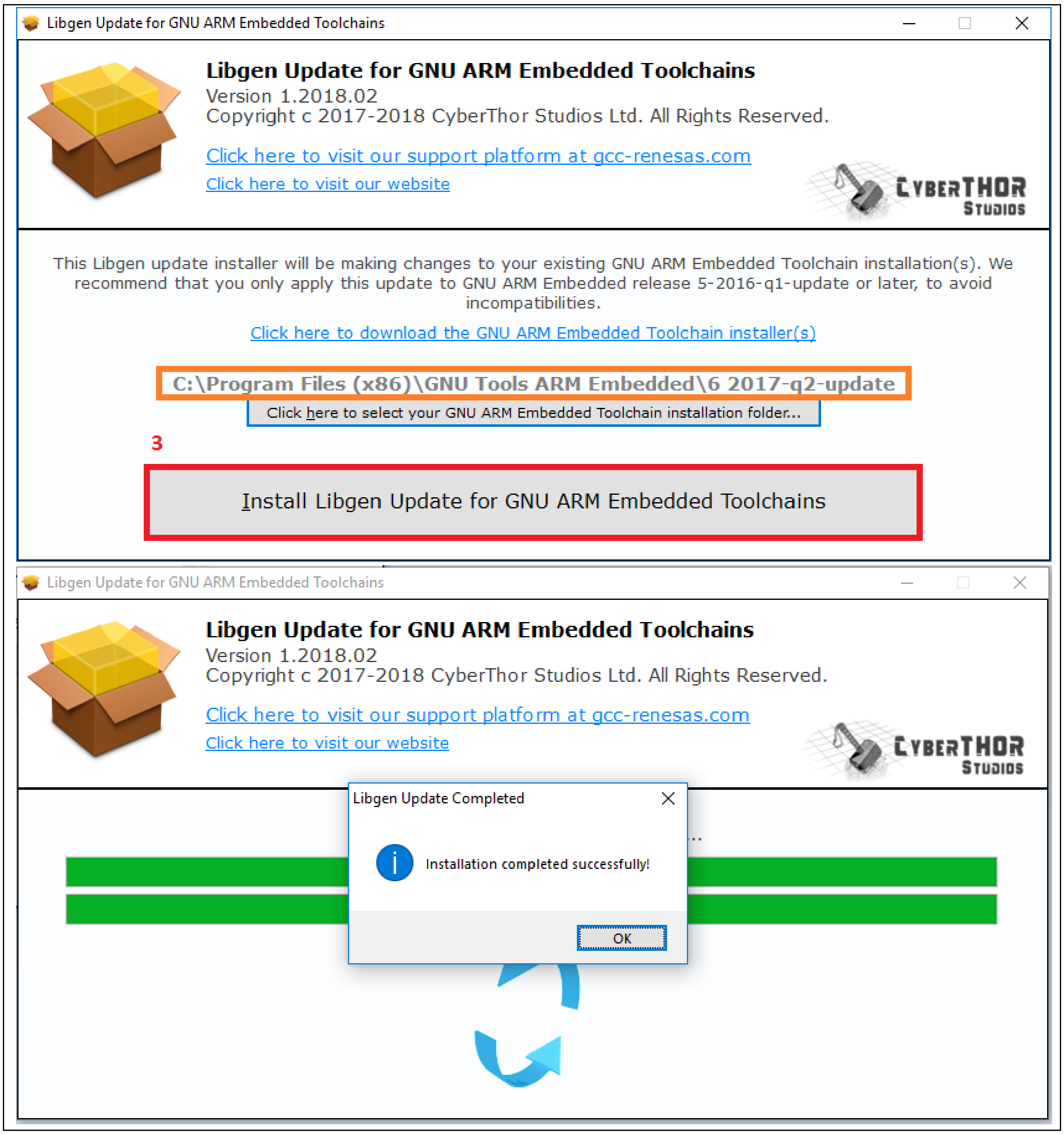
C:¥...¥GNU Tools ARM Embedded¥6 2017-q2-update

Adobe
Atmel
Citrix
Common Files
DAUM
e-stamp_pro
ffftp
GNU Tools ARM Embedded
2 6 2017-q2-update
Google

New folder OK Cancel

Figure 2-24. Libgen Update for GNU ARM Embedded: Select Toolchain Folder

3. Next, click **Install Libgen Update for GNU ARM Embedded Toolchains**.



Libgen Update for GNU ARM Embedded Toolchains

Libgen Update for GNU ARM Embedded Toolchains
Version 1.2018.02
Copyright c 2017-2018 CyberThor Studios Ltd. All Rights Reserved.

Click here to visit our support platform at gcc-renesas.com
Click here to visit our website

This Libgen update installer will be making changes to your existing GNU ARM Embedded Toolchain installation(s). We recommend that you only apply this update to GNU ARM Embedded release 5-2016-q1-update or later, to avoid incompatibilities.

Click here to download the GNU ARM Embedded Toolchain installer(s)

C:\Program Files (x86)\GNU Tools ARM Embedded\6 2017-q2-update

Click here to select your GNU ARM Embedded Toolchain installation folder...

3

Install Libgen Update for GNU ARM Embedded Toolchains

Libgen Update for GNU ARM Embedded Toolchains

Libgen Update for GNU ARM Embedded Toolchains
Version 1.2018.02
Copyright c 2017-2018 CyberThor Studios Ltd. All Rights Reserved.

Click here to visit our support platform at gcc-renesas.com
Click here to visit our website

Libgen Update Completed

Installation completed successfully!

OK

Figure 2-25. Libgen Update for GNU ARM Embedded: Update Toolchain

If you want to update another toolchain, download **Libgen Update for GNU ARM Embedded** separately and run it (refer to Chapter 1.4).

2.4 Uninstallation of the e^2 studio IDE

The e^2 studio IDE can be uninstalled by following the usual steps to uninstall a program in a Windows OS.

1. Search for **Apps & features** in the Windows Search Box. Click on the search result to go to **Apps & features**.

2. From the currently installed programs list, choose 'e2 studio' and click the **Uninstall** button.

3. Click the **Uninstall** button again to confirm the deletion of e^2 studio.

At the end of the uninstallation, the e^2 studio IDE will be deleted from the installed location and Windows shortcut menus will be removed.

2.5 Major Version Upgrade for the e^2 studio IDE

The **Check for Updates** operation does not work for major version upgrades (a major digit version number increase, for example from version $\boxed{6}$.3.0 to version $\boxed{7}$.0.0), you need to use the e^2 studio installer for major upgrades. Please note that you should not overwrite the existing installation. Before installing the new version, you must uninstall the old one. However, to keep both the old and new IDE versions, you can install the new version to a different location.

2.6 Update e^2 studio plugins

The e^2 studio IDE supports online and offline updates for minor version updates (i.e. minor digit number increases, for example going from version 7.$\boxed{3.0}$ to version 7.$\boxed{4.0}$). This allows the available software components to be updated over the internet or without an internet connection.

2.6.1 Online Minor Version Update

This chapter shows how to perform a minor version update online:

1. From the **Help** menu, click **Check for Updates** to display the **Available Updates** panel.

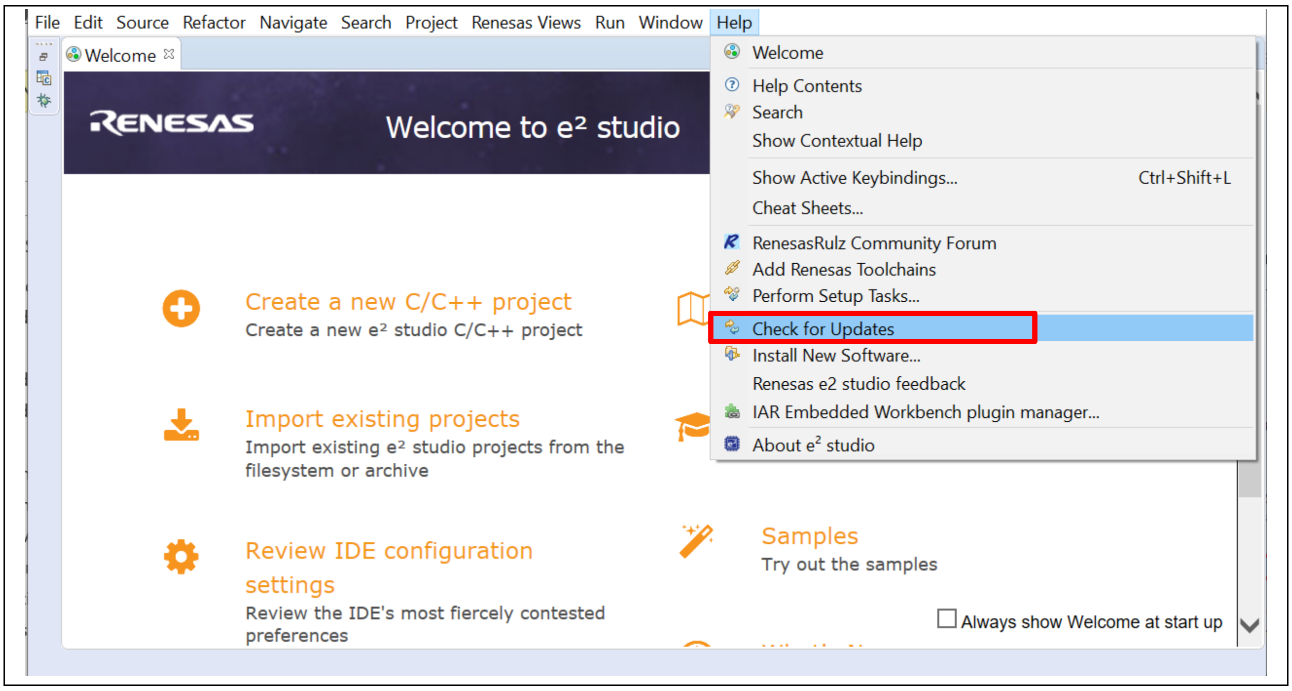
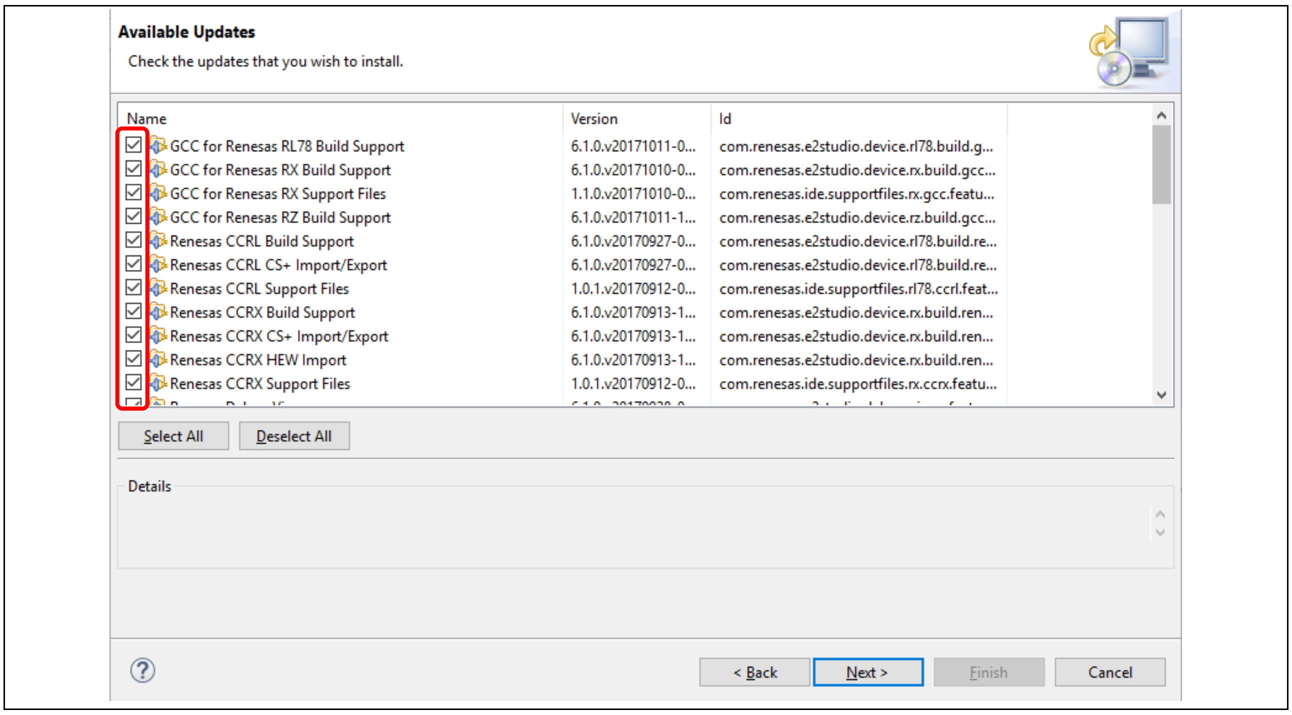

File
Edit
Source
Refactor
Navigate
Search
Project
Renesas Views
Run
Window
Help
Welcome
RENESAS
Welcome to e² studio
Create a new C/C++ project
Create a new e² studio C/C++ project
Import existing projects
Import existing e² studio projects from the filesystem or archive
Review IDE configuration settings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

Figure 2-26. Check for Updates Menu

2. All the software components are selected by default in the **Available Updates** panel. This allows you to update them all to the latest version (an example is shown in Figure 2-2). Click the **Next** button to proceed.



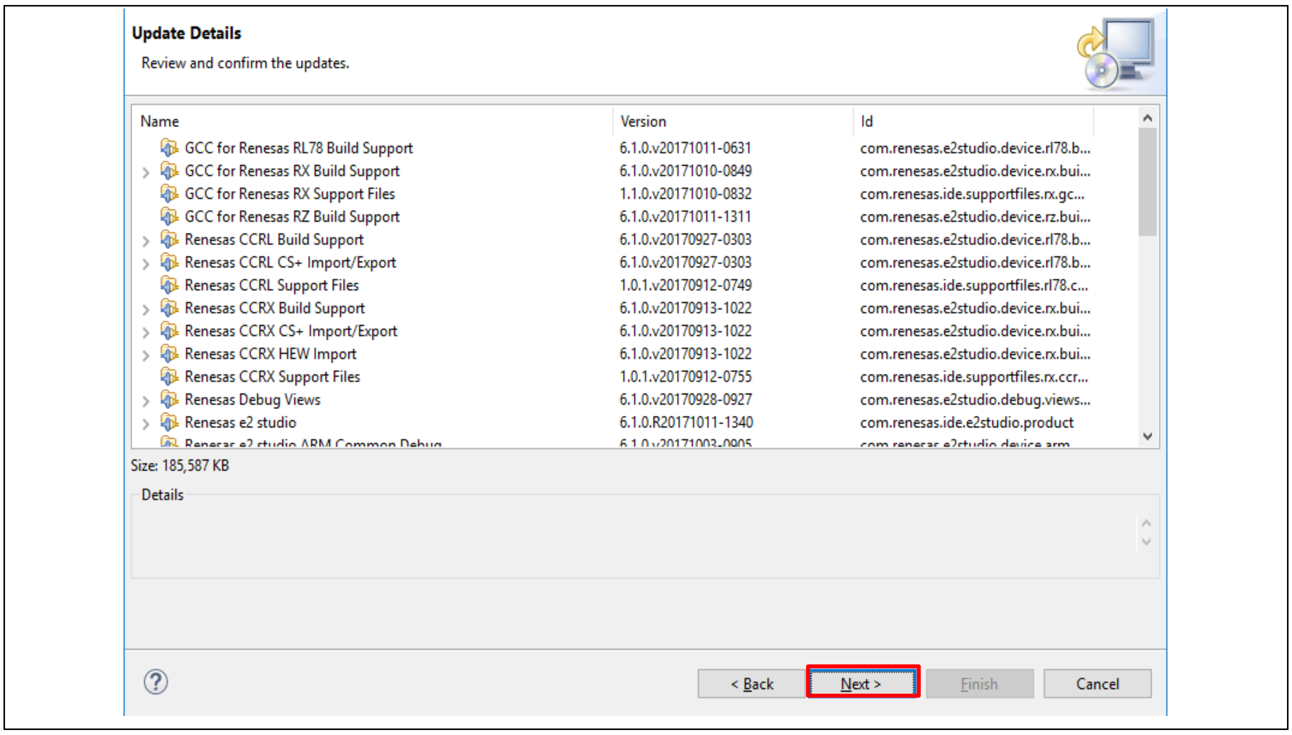
Available Updates

Check the updates that you wish to install.

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

Figure 2-27. e² studio – Available Updates Panel (1/3)

3. Select the **Next** button to continue the update.



Update Details

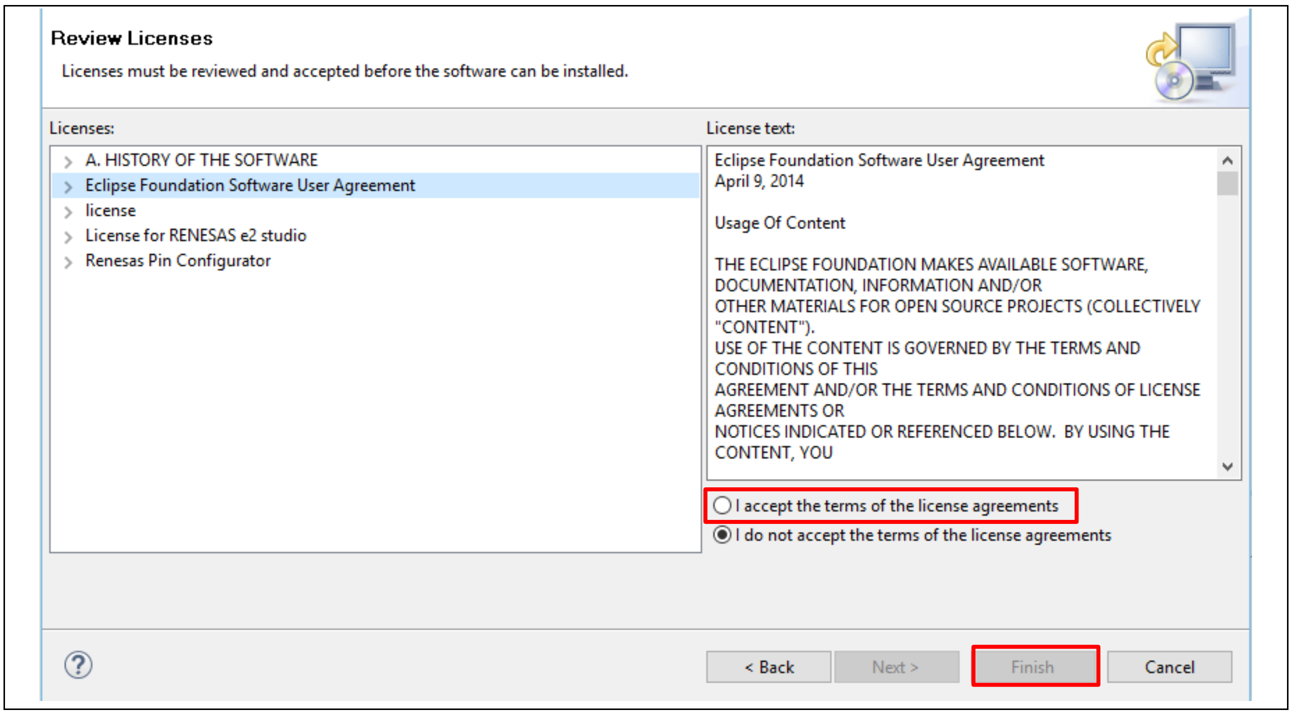
Review and confirm the updates.

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

Size: 185,587 KB

Figure 2-28. e² studio – Available Updates Panel (2/3)

4. Read and accept the software license agreement. Click the **Finish** button to complete the update.



Review Licenses

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

Licenses:

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

License text:

Eclipse Foundation Software User Agreement
April 9, 2014

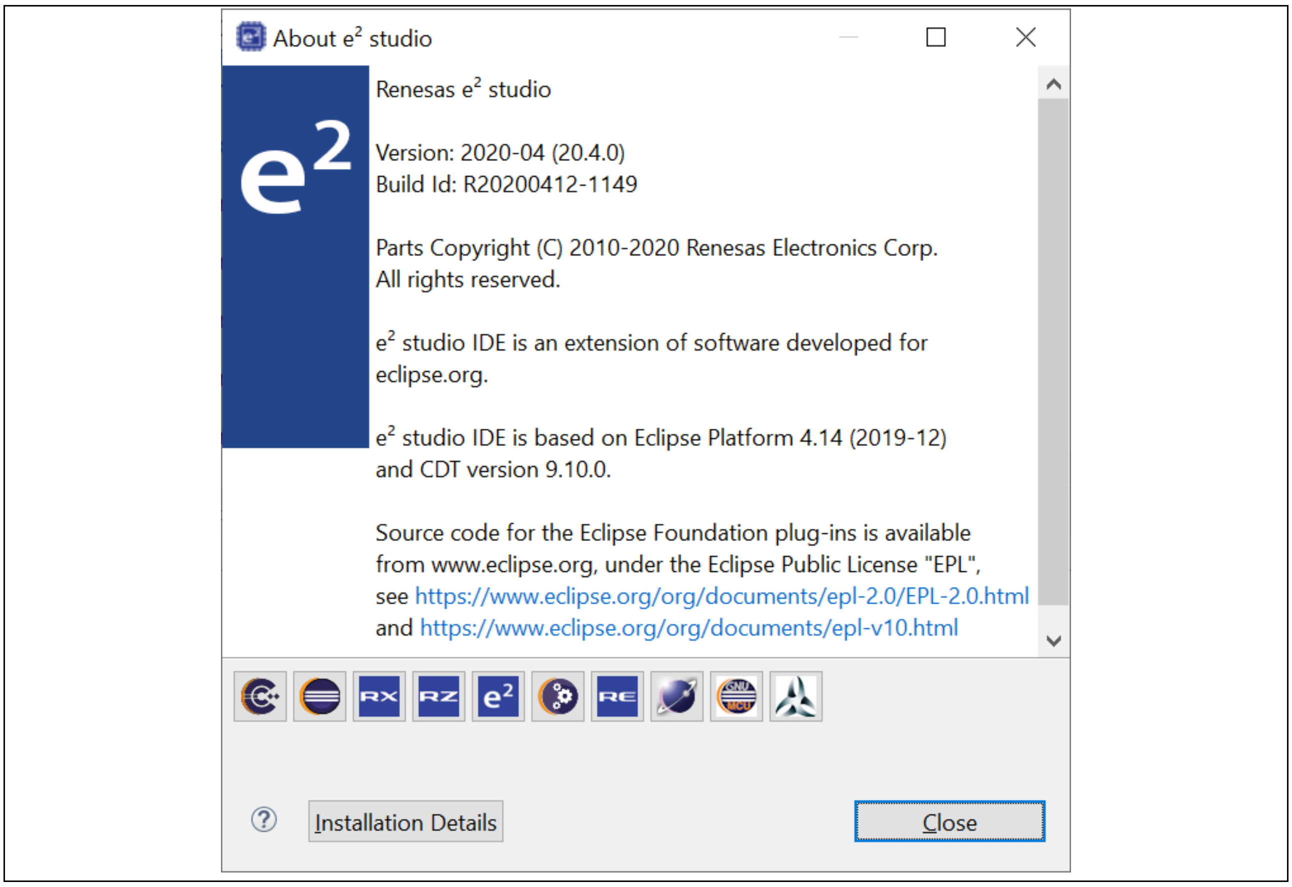
Usage Of Content

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

I accept the terms of the license agreements [✓ I do not accept the terms of the license agreements]

Figure 2-29. e^2 studio – Available Updates Panel (3/3)

5. Click the **Help** → **About e^2 studio** menu option to check the version number and verify the update.



About e^2 studio

Renesas e^2 studio

Version: 2020-04 (20.4.0)
Build Id: R20200412-1149

Parts Copyright (C) 2010-2020 Renesas Electronics Corp.
All rights reserved.

e^2 studio IDE is an extension of software developed for eclipse.org.

e^2 studio IDE is based on Eclipse Platform 4.14 (2019-12) and CDT version 9.10.0.

Source code for the Eclipse Foundation plug-ins is available from www.eclipse.org, under the Eclipse Public License "EPL", see https://www.eclipse.org/org/documents/epl-2.0/EPL-2.0.html and https://www.eclipse.org/org/documents/epl-v10.html

Figure 2-30. e^2 studio – About e^2 studio Panel

2.6.2 Offline Minor Version Update

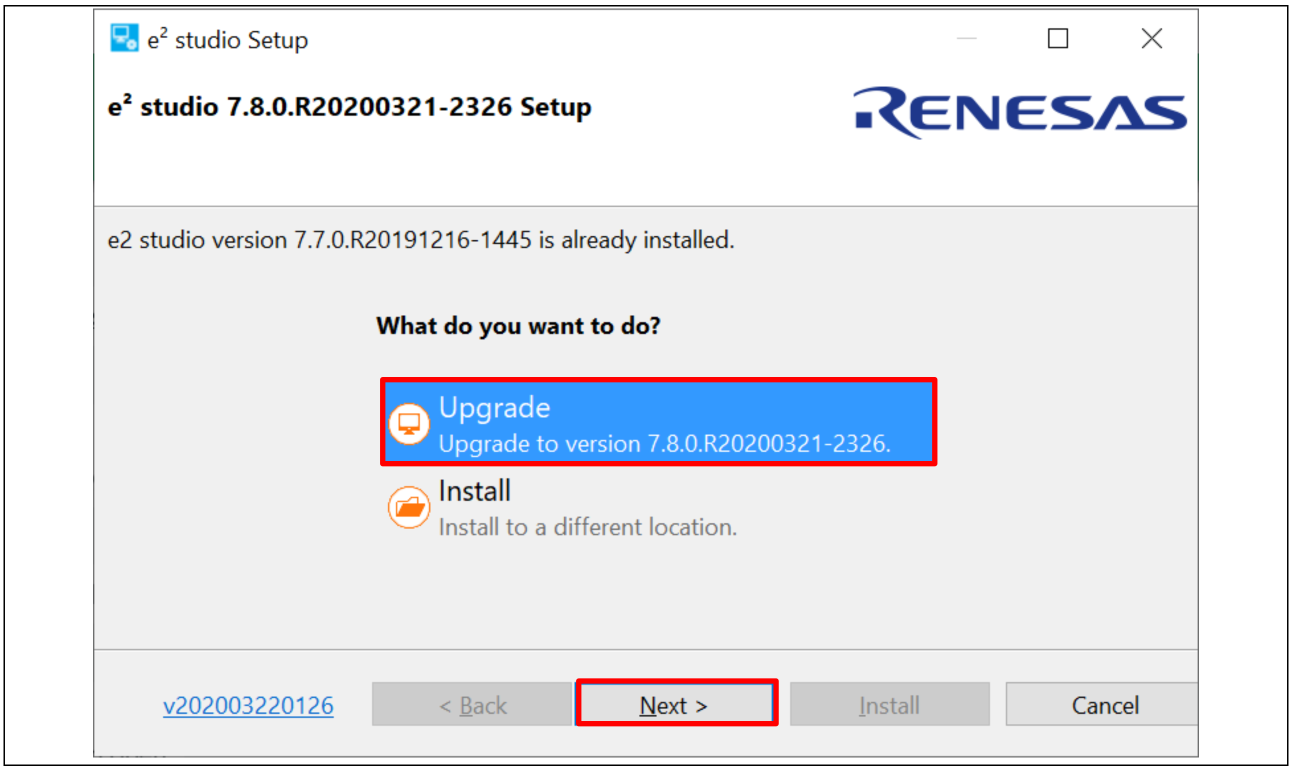
This chapter shows how to update e^2 studio using the update function of the installer.

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

    Note: Offline update using the 'Differential Update Program' is only applicable for e^2 studio version 3.X and below.

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

3. Click **Upgrade** followed by **Next** to start the update.



e^2 studio Setup

e^2 studio 7.8.0.R20200321-2326 Setup

e2 studio version 7.7.0.R20191216-1445 is already installed.

What do you want to do?

Upgrade
Upgrade to version 7.8.0.R20200321-2326.

Install
Install to a different location.

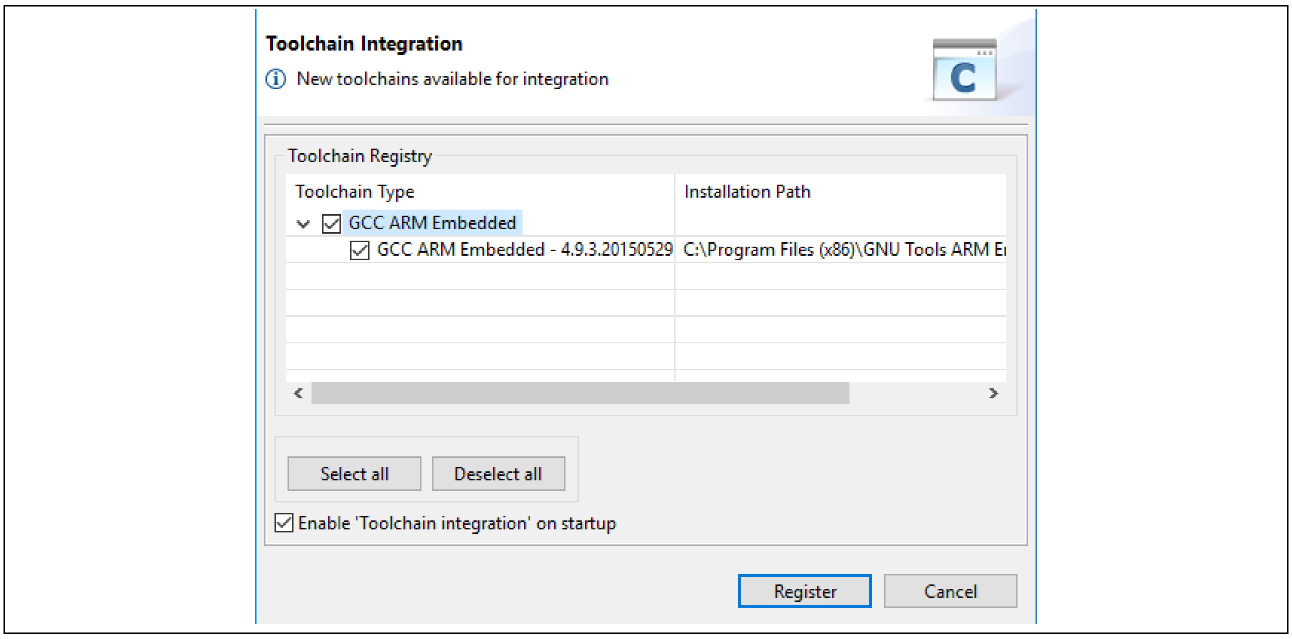
Figure 2-31. Upgrade e^2 studio from Offline Installer

4. Follow the steps shown in Chapter 2.1 **Installation of e^2 studio IDE**. Step 2 **Install Folder** will be skipped since the upgrade option will use the same destination folder as the existing e^2 studio installation.

2.7 Register Toolchain to e^2 studio

The e^2 studio installer can install toolchains (such as GCC ARM Embedded) automatically during e^2 studio installation with a valid Internet connection. However, in situations where an Internet connection is not available during e^2 studio installation, toolchains that are needed can be installed later by using installer files from the web site and then registered to e^2 studio. For the toolchain download site for RZ, please refer to chapter 1.3.

e^2 studio scans for installed toolchains on start up. After a new toolchain installation, e^2 studio will pop up a toolchain integration window to allow for toolchain selection.


Toolchain Integration

New toolchains available for integration

Toolchain Registry

Toolchain Type	Installation Path
☑ GCC ARM Embedded	
☑ GCC ARM Embedded - 4.9.3.20150529	C:\Program Files (x86)\GNU Tools ARM E

☑ Enable 'Toolchain integration' on startup

Figure 2-32. e^2 studio Toolchain Integration

To check for installed compilers, click the **Integrate non-integrated toolchains** icon from the toolbar, or click **Help → Add Renesas Toolchains** to open Renesas Toolchain Management as shown below.
Check the desired toolchain to integrate it with e^2 studio.

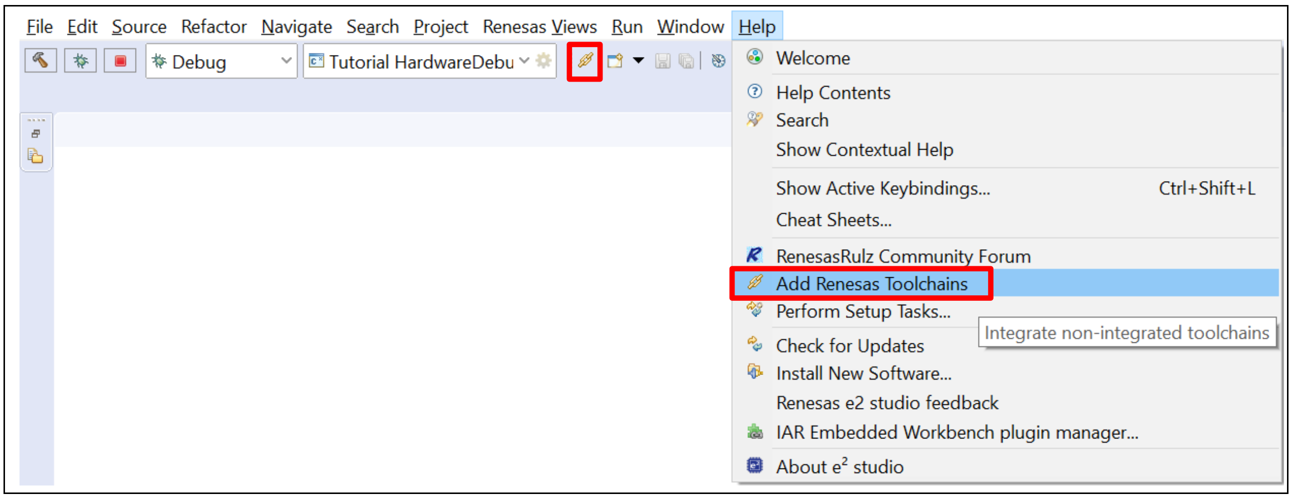
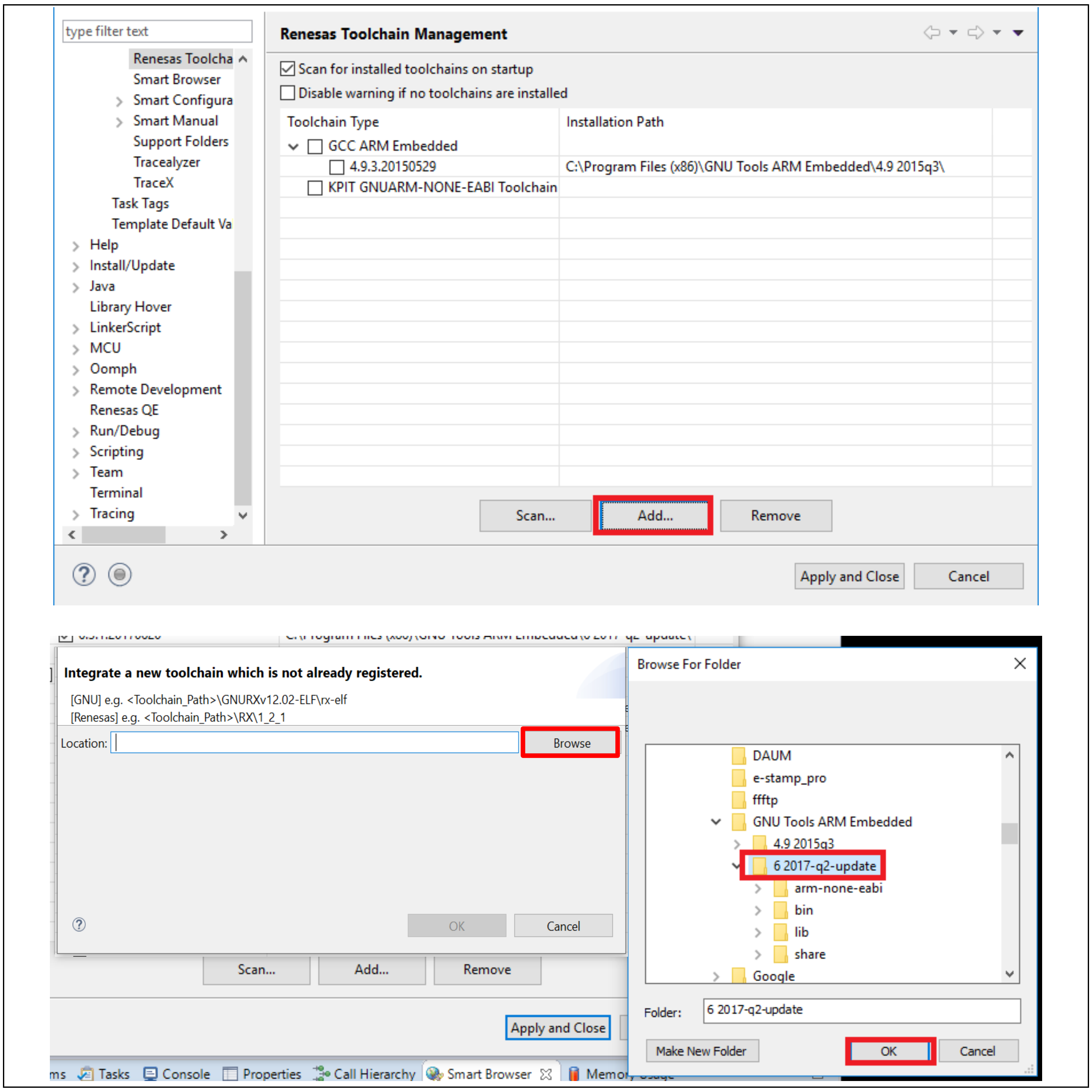

File Edit Source Refactor Navigate Search Project Renesas Views Run Window Help
Debug
Tutorial HardwareDebu
Welcome
Help Contents
Search
Show Contextual Help
Show Active Keybindings... Ctrl+Shift+L
Cheat Sheets...
RenesasRulz Community Forum
Add Renesas Toolchains
Perform Setup Tasks...
Integrate non-integrated toolchains
Check for Updates
Install New Software...
Renesas e2 studio feedback
IAR Embedded Workbench plugin manager...
About e² studio

Figure 2-33. e^2 studio Add Renesas Toolchains

If the desired toolchain is not listed, Click **Add…** and specify the location where it is installed.



Renesas Toolchain Management

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

Toolchain Type	Installation Path
☐ GCC ARM Embedded	
☐ 4.9.3.20150529	C:\Program Files (x86)\GNU Tools ARM Embedded\4.9 2015q3\
☐ KPIT GNUARM-NONE-EABI Toolchain	

Integrate a new toolchain which is not already registered.

[GNU] e.g. <Toolchain_Path>\GNURXv12.02-ELF\rx-elf
[Renesas] e.g. <Toolchain_Path>\RX\1_2_1

Location:

Browse For Folder

- DAUM
- e-stamp_pro
- ffftp
- GNU Tools ARM Embedded
  - 4.9 2015q3
  - 6 2017-q2-update
    - arm-none-eabi
    - bin
    - lib
    - share
- Google

Folder: 6 2017-q2-update

Figure 2-34. Toolchain Management: Register New Toolchain

3. Import / Generate Project

This chapter describes how to import existing projects (such as sample code in software packages) and how to generate a new project.

Note: Multi-byte characters cannot be used for the e^2 studio installation folder name, project name, project folder, or source file names.

3.1 Import Existing Project

Renesas provides various useful sample projects in the RZ device software package. These sample projects can easily be imported into e^2 studio. You can also import your own projects this way.

1. Download the software package.
2. Open the **Import** dialog with a click on “Import projects…” link in **Project Explorer** (if there is no projects in the workspace) or select the menu **File → Import**.

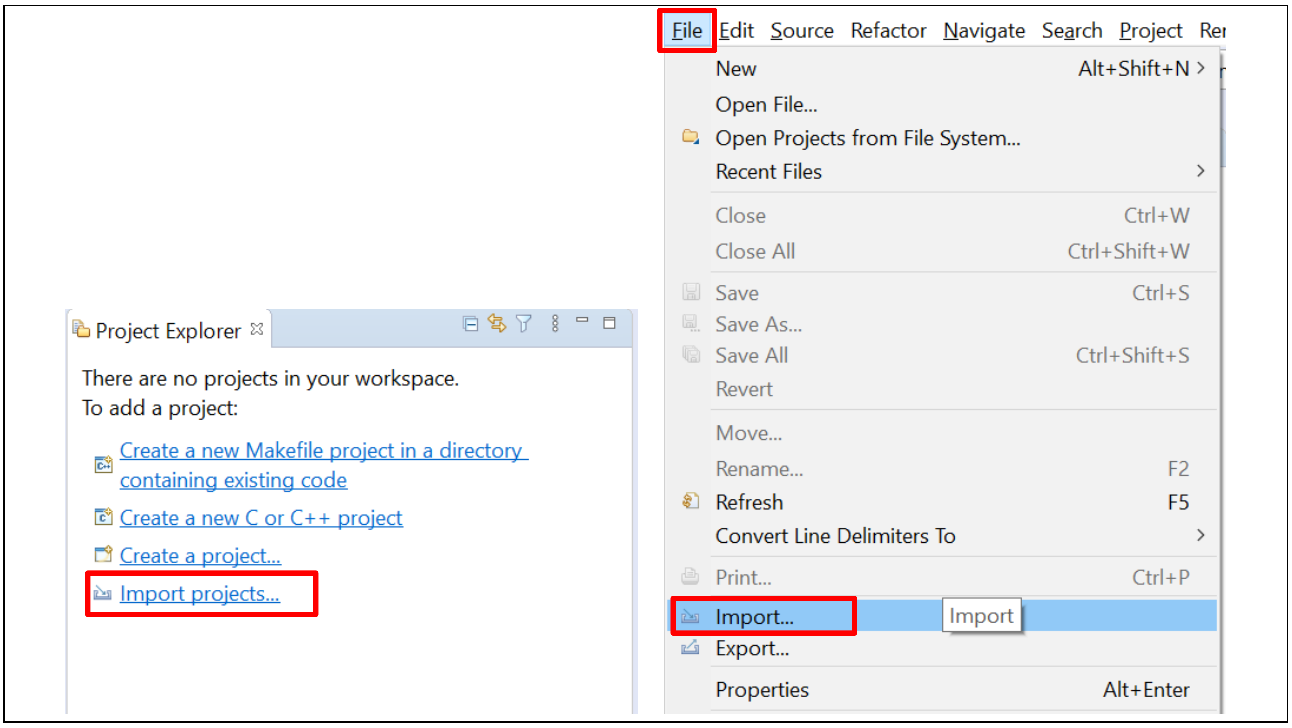
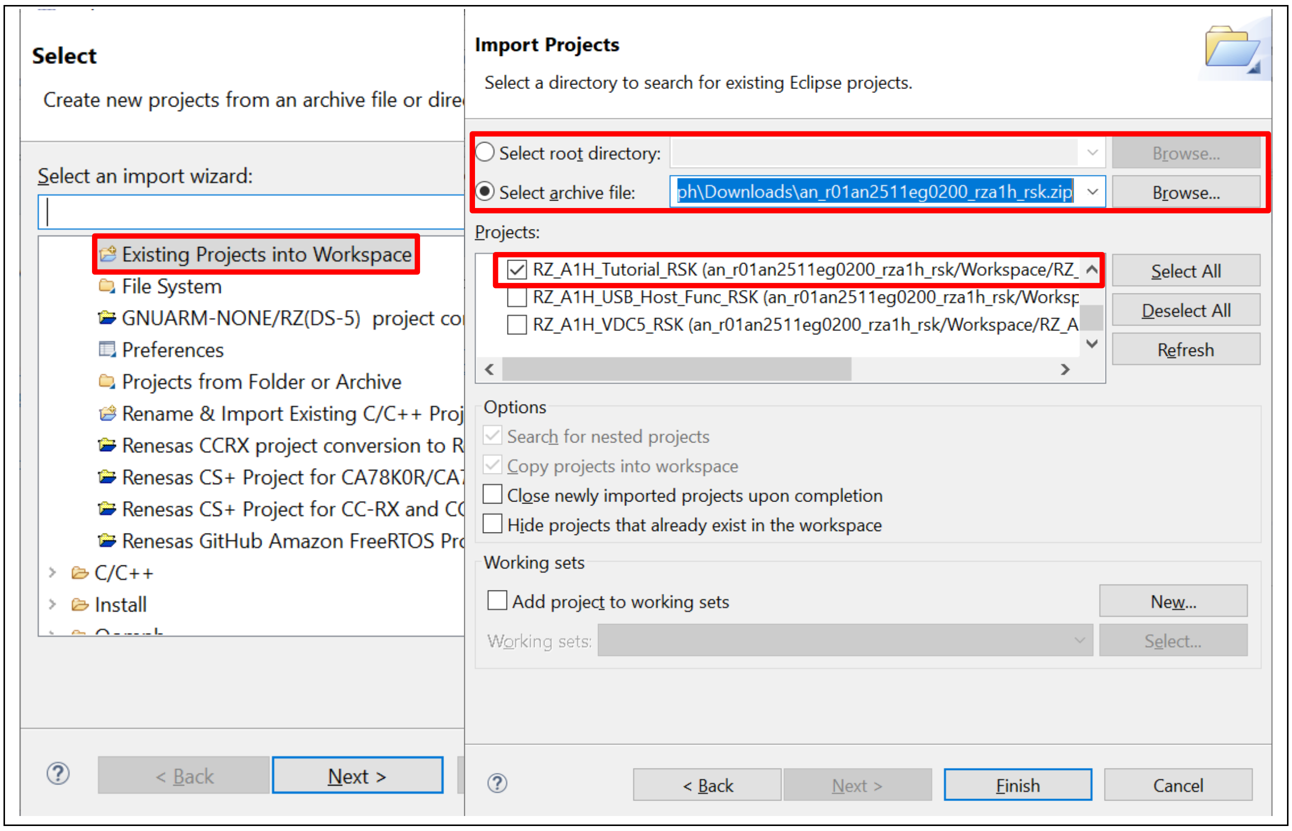


Figure 3-1. Import Menu

3. Select **General → Existing Projects into Workspace** and click **Next**.

4. Select the folder or project archive file then click **Browse** and select the project folder in the workspace.


Select
Create new projects from an archive file or dire
Select an import wizard:
Existing Projects into Workspace
File System
GNUARM-NONE/RZ(DS-5) project co
Preferences
Projects from Folder or Archive
Rename & Import Existing C/C++ Proj
Renesas CCRX project conversion to R
Renesas CS+ Project for CA78K0R/CA7
Renesas CS+ Project for CC-RX and CC
Renesas GitHub Amazon FreeRTOS Pr
C/C++
Install
< Back
Next >
Import Projects
Select a directory to search for existing Eclipse projects.
Select root directory:
Browse...
Select archive file:
Browse...
Projects:
Select All
Deselect All
Refresh
Options
Working sets
Add project to working sets
New...
Working sets:
Select...
< Back
Next >
Finish
Cancel

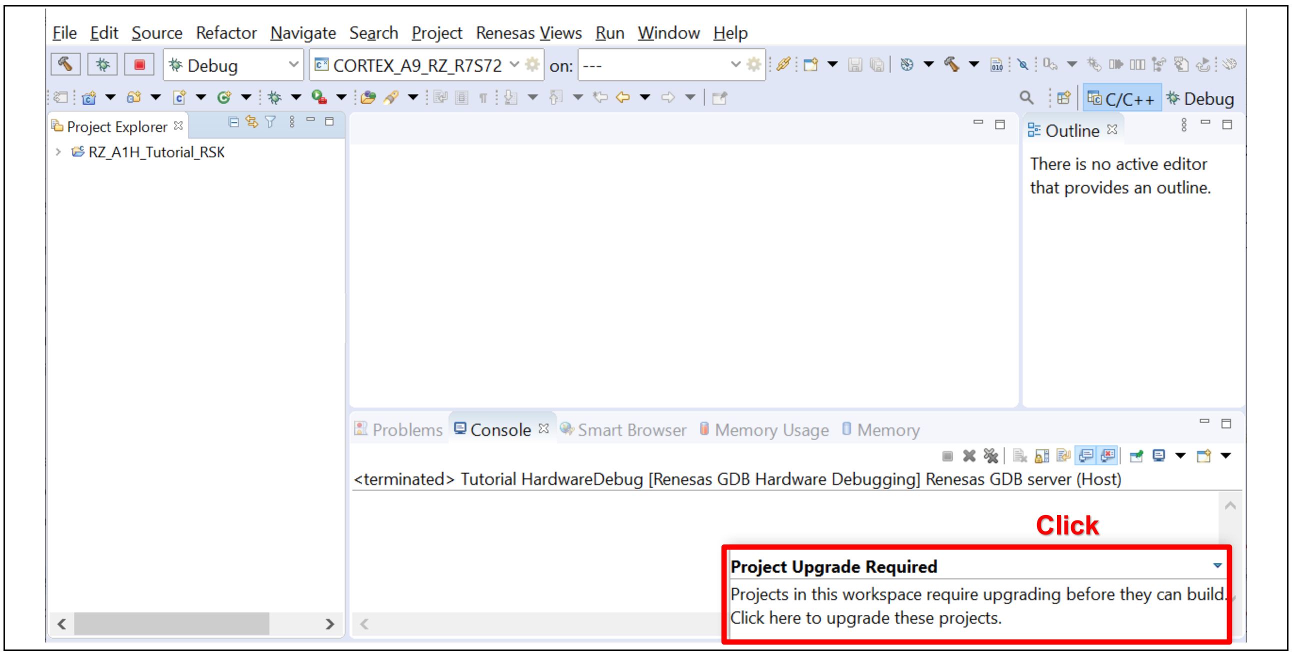
Select archive file: ph\Downloads\an_r01an2511eg0200_rza1h_rsk.zip

☑ RZ_A1H_Tutorial_RSK (an_r01an2511eg0200_rza1h_rsk/Workspace/RZ_
☐ RZ_A1H_USB_Host_Func_RSK (an_r01an2511eg0200_rza1h_rsk/Worksp
☐ RZ_A1H_VDC5_RSK (an_r01an2511eg0200_rza1h_rsk/Workspace/RZ_A

☑ Search for nested projects
☑ Copy projects into workspace
☐ Close newly imported projects upon completion
☐ Hide projects that already exist in the workspace

Figure 3-2 Select Folder or Archive File Included Project Files

5. Projects generated by legacy e^2 studio (before V6.0) will need to be upgraded.
   Click the **Project Upgrade Required** pop-up message (continued on the next page).



There is no active editor that provides an outline.

<terminated> Tutorial HardwareDebug [Renesas GDB Hardware Debugging] Renesas GDB server (Host)

Project Upgrade Required
Projects in this workspace require upgrading before they can build.
Click here to upgrade these projects.

Figure 3-3. Project Upgrade

Alternatively, run **Upgrade Legacy e2 studio Projects…** from the project right-button context menu.

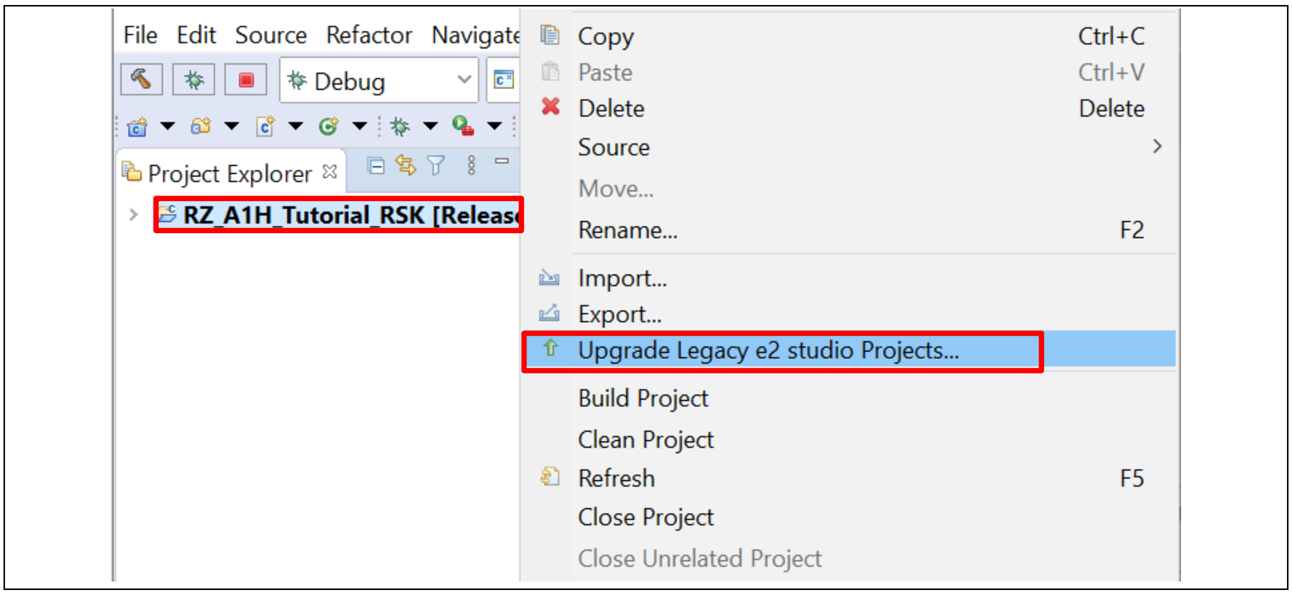
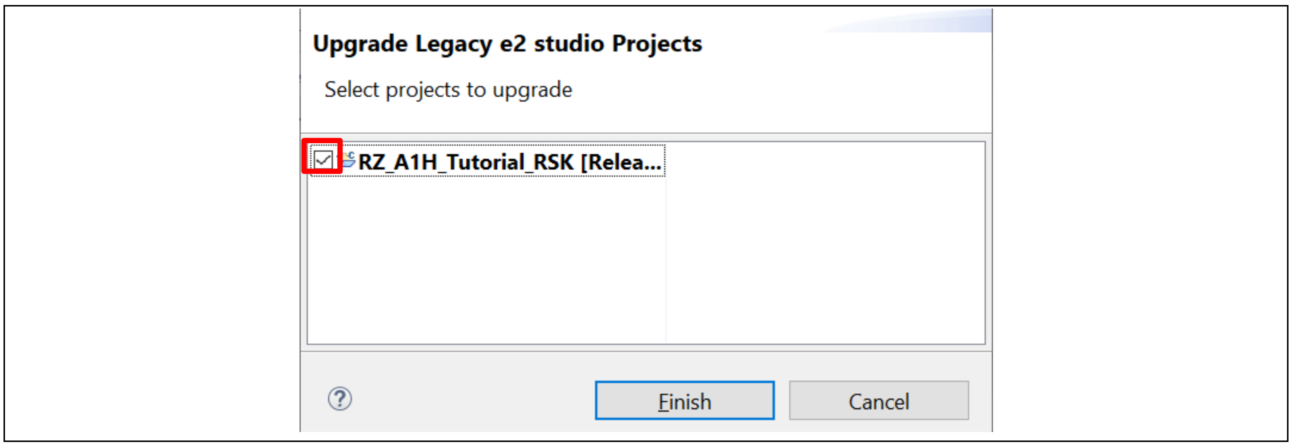


Figure 3-4. Manual Project Upgrade

6. Select the projects to upgrade and click **Finish** (Figure 3-5).


Upgrade Legacy e2 studio Projects

Select projects to upgrade

Figure 3-5. Migration Required Dialog Box

3.2 New Project Generation

When creating a new project, the method of creating the project differs between RZ/A2M and other RZ groups. This chapter explains how to generate a project for each device.

3.2.1 Project Generation of RZ/A2M

This chapter illustrates an example when using the RZ/A2M Evaluation Board Kit. The RZ/A2M project type can be selected from 'Loader Project' or 'Application Project'. On the RZ/A2M, both types of project need to be built and downloaded when debugging.

A 'Loader Project' is a project for generating a loader program, and an 'Application Project' is a project for creating a user application program. A 'Loader Project' sets the serial flash memory of the RZ/A2M Evaluation Board Kit so that it can be accessed at high speed, and then branches to the 'Application Project' (if the loader program has not been downloaded, the application program cannot be executed).

When using it on your own system, change the program according to the specifications of the serial flash memory being used.

To create a new project, launch the e^2 studio IDE from the Windows start menu and specify a workspace directory.

1. Click **File → New → C/C++ Project** to open the **New Project Creation Wizard**.

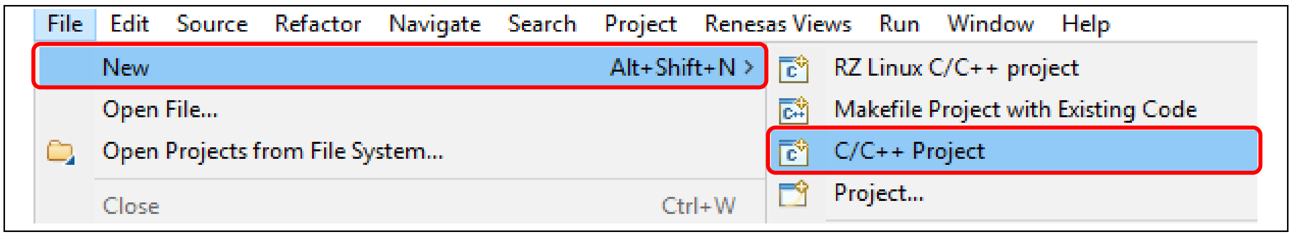
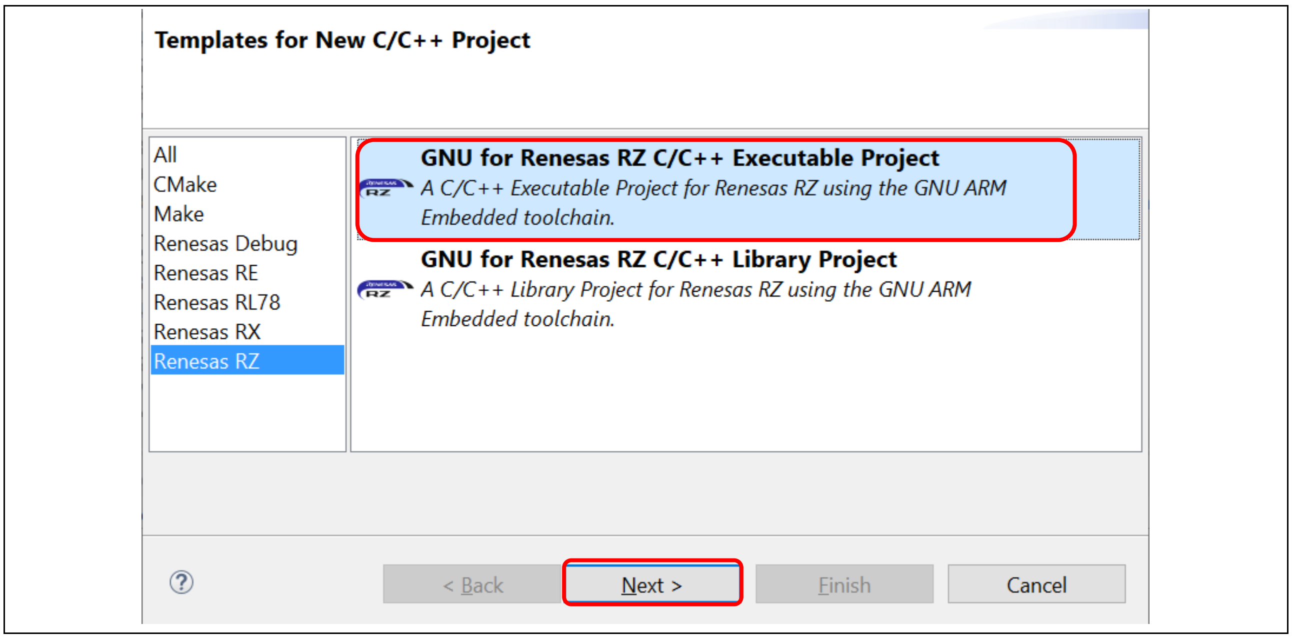


Figure 3-6. Open New Project Creation Wizard

2. Select the template for the new project (for example Renesas RZ: **GCC for Renesas RZ C/C++ Executable Project**). Click **Next** to proceed.


Templates for New C/C++ Project

All
CMake
Make
Renesas Debug
Renesas RE
Renesas RL78
Renesas RX
Renesas RZ

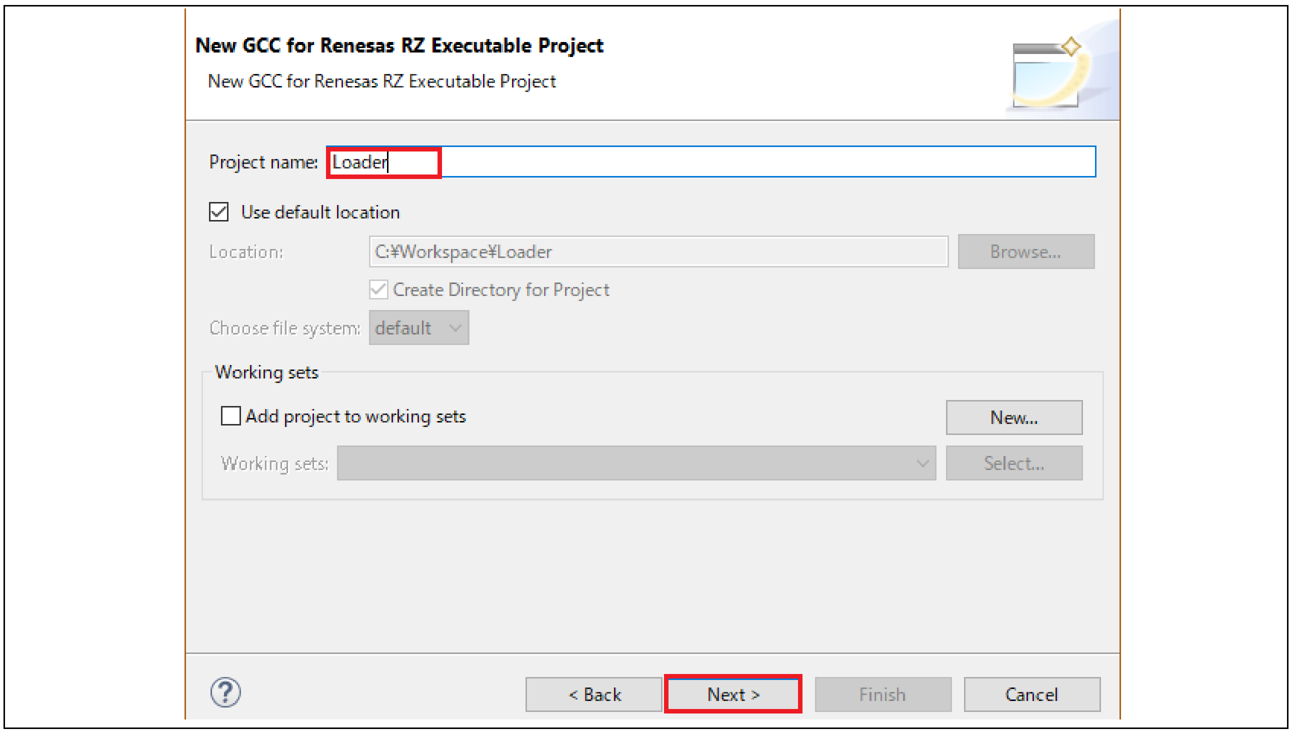
GNU for Renesas RZ C/C++ Executable Project
A C/C++ Executable Project for Renesas RZ using the GNU ARM Embedded toolchain.

GNU for Renesas RZ C/C++ Library Project
A C/C++ Library Project for Renesas RZ using the GNU ARM Embedded toolchain.

< Back | Next > | Finish | Cancel

Figure 3-7. Select Project Template

3. First, create a loader project. Enter the project name ('Loader'). Click **Next** to proceed.


New GCC for Renesas RZ Executable Project
New GCC for Renesas RZ Executable Project

Project name: Loader

☑ Use default location
Location: C:¥Workspace¥Loader Browse...
☑ Create Directory for Project
Choose file system: default

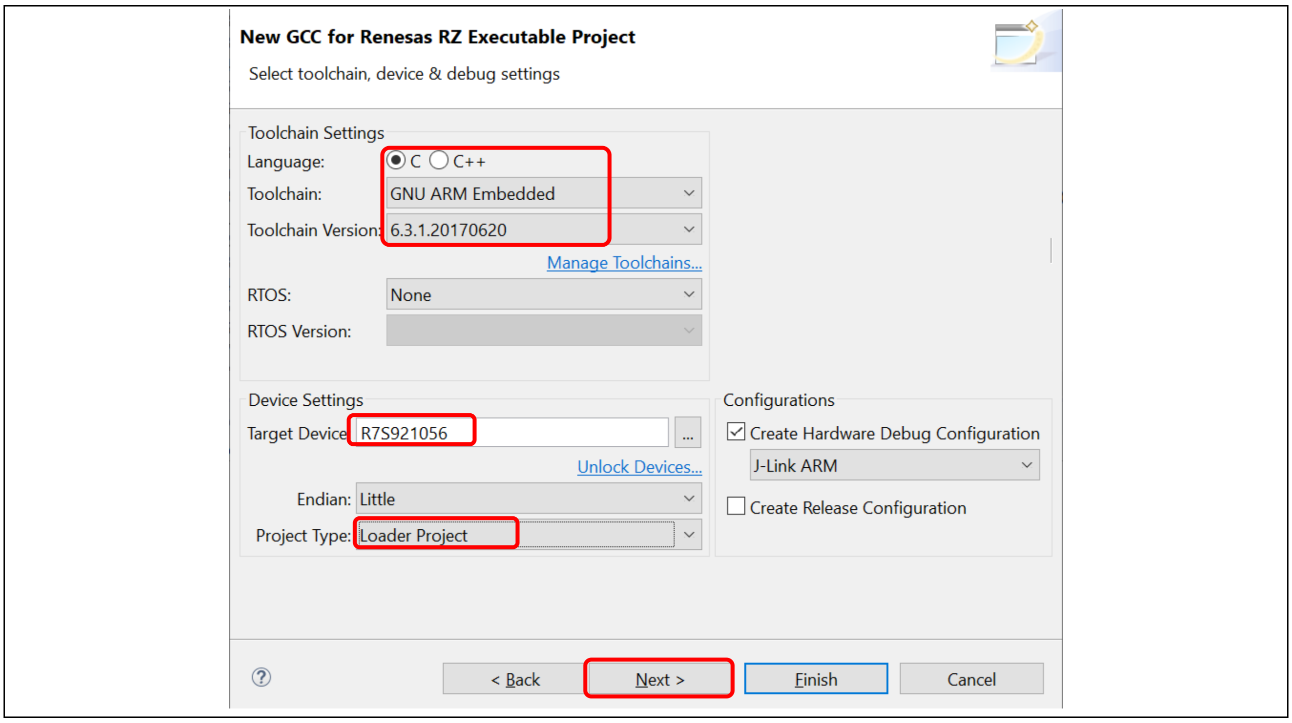
Working sets
☐ Add project to working sets New...
Working sets: Select...

< Back | Next > | Finish | Cancel

Figure 3-8. Specify Project Name

4. Select the following settings:
    - Language: 'C'
    - Toolchain: 'GCC ARM Embedded'
    - Toolchain Version: '6.3.1.20170620'
    - Target Device: 'R7S921053'
    - Project Type: 'Loader Project'

   Select **Create Hardware Debug Configuration** and then **J-Link ARM**. Click **Next** to proceed.



New GCC for Renesas RZ Executable Project

Select toolchain, device & debug settings

Toolchain Settings

Language: [✓ C] C++

Toolchain: GNU ARM Embedded

Toolchain Version: 6.3.1.20170620

Manage Toolchains...

RTOS: None

RTOS Version:

Device Settings

Target Device: R7S921056

Unlock Devices...

Endian: Little

Project Type: Loader Project

Configurations

[✓] Create Hardware Debug Configuration

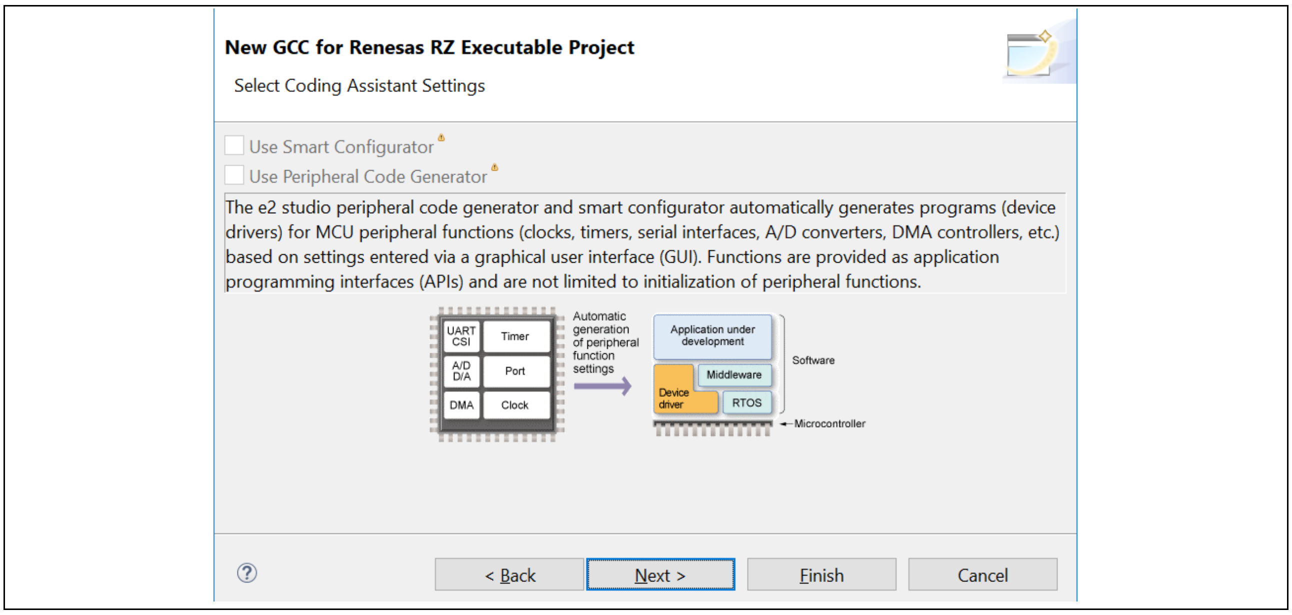
J-Link ARM

[ ] Create Release Configuration

< Back | Next > | Finish | Cancel

Figure 3-9. Select Toolchain, Device & Debug Settings: e.g. RZ/A2M Renesas Starter Kit

5. Click **Next** to proceed because Loader Projects do not support Smart Configurator.



New GCC for Renesas RZ Executable Project

Select Coding Assistant Settings

[ ] Use Smart Configurator

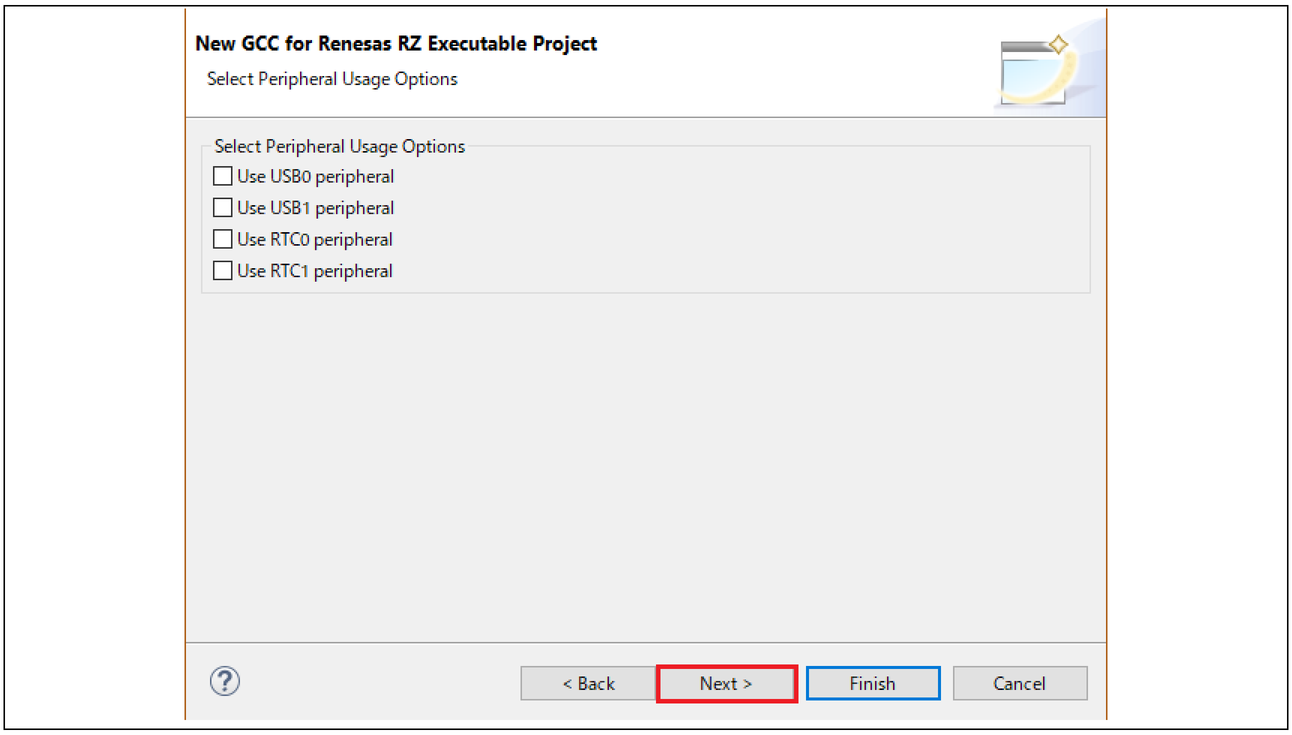
[ ] Use Peripheral Code Generator

The e2 studio peripheral code generator and smart configurator automatically generates programs (device drivers) for MCU peripheral functions (clocks, timers, serial interfaces, A/D converters, DMA controllers, etc.) based on settings entered via a graphical user interface (GUI). Functions are provided as application programming interfaces (APIs) and are not limited to initialization of peripheral functions.

< Back | Next > | Finish | Cancel

Figure 3-10. Select Coding Assistant Settings

6. In the initial settings, USB and RTC are not used. In 'Loader Project', click **Next** to proceed.



New GCC for Renesas RZ Executable Project

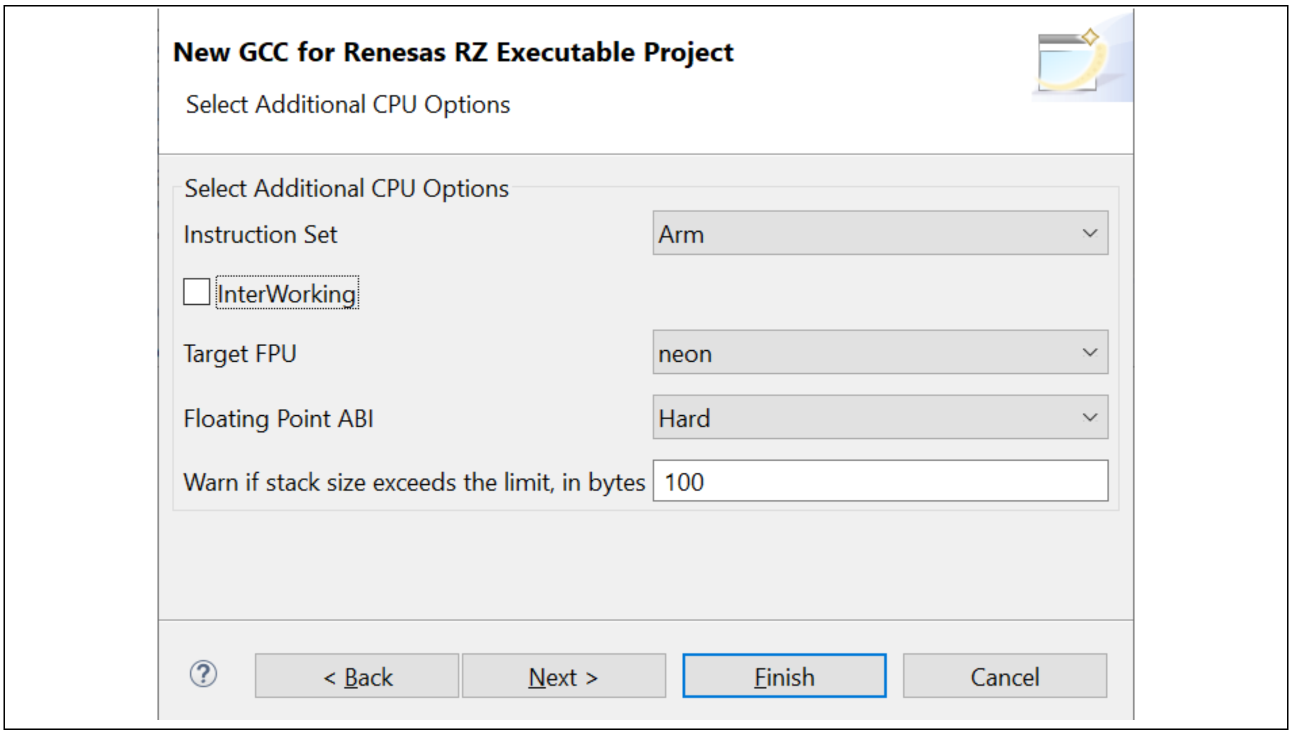
Select Peripheral Usage Options

Select Peripheral Usage Options

- [ ] Use USB0 peripheral
- [ ] Use USB1 peripheral
- [ ] Use RTC0 peripheral
- [ ] Use RTC1 peripheral

Figure 3-11. Select Peripheral Usage Options

7. Keep default settings for additional CPU options and click **Next** to proceed.



New GCC for Renesas RZ Executable Project

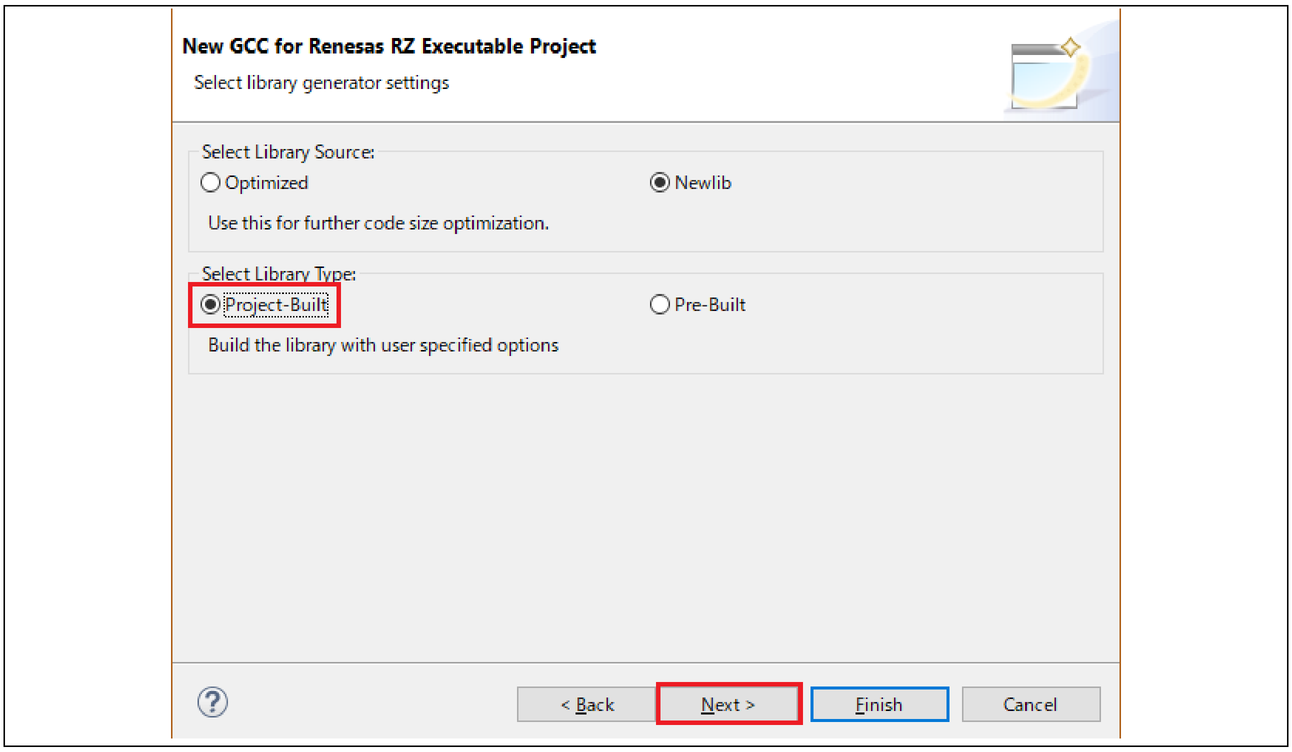
Select Additional CPU Options

Select Additional CPU Options

Instruction Set	Arm
[ ] InterWorking	
Target FPU	neon
Floating Point ABI	Hard
Warn if stack size exceeds the limit, in bytes	100

Figure 3-12. Select Additional CPU Options

8. Select 'Project-Built' and click **Next** to proceed.


New GCC for Renesas RZ Executable Project

Select library generator settings

Select Library Source:

Optimized [✓ Newlib]

Use this for further code size optimization.

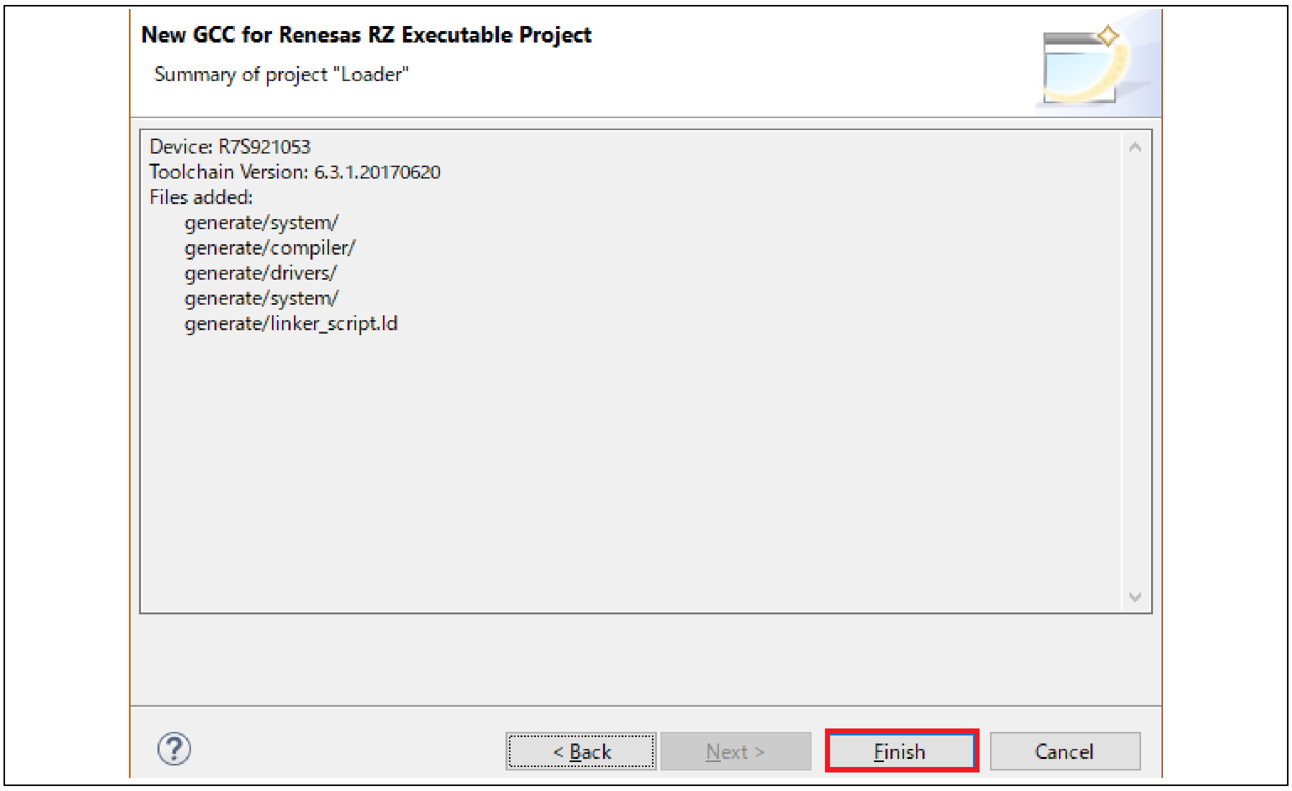
Select Library Type:

[✓ Project-Built] Pre-Built

Build the library with user specified options

Figure 3-13. Library Generator Settings: Select Project Built

9. A project summary is displayed. Click **Finish** to generate the project.

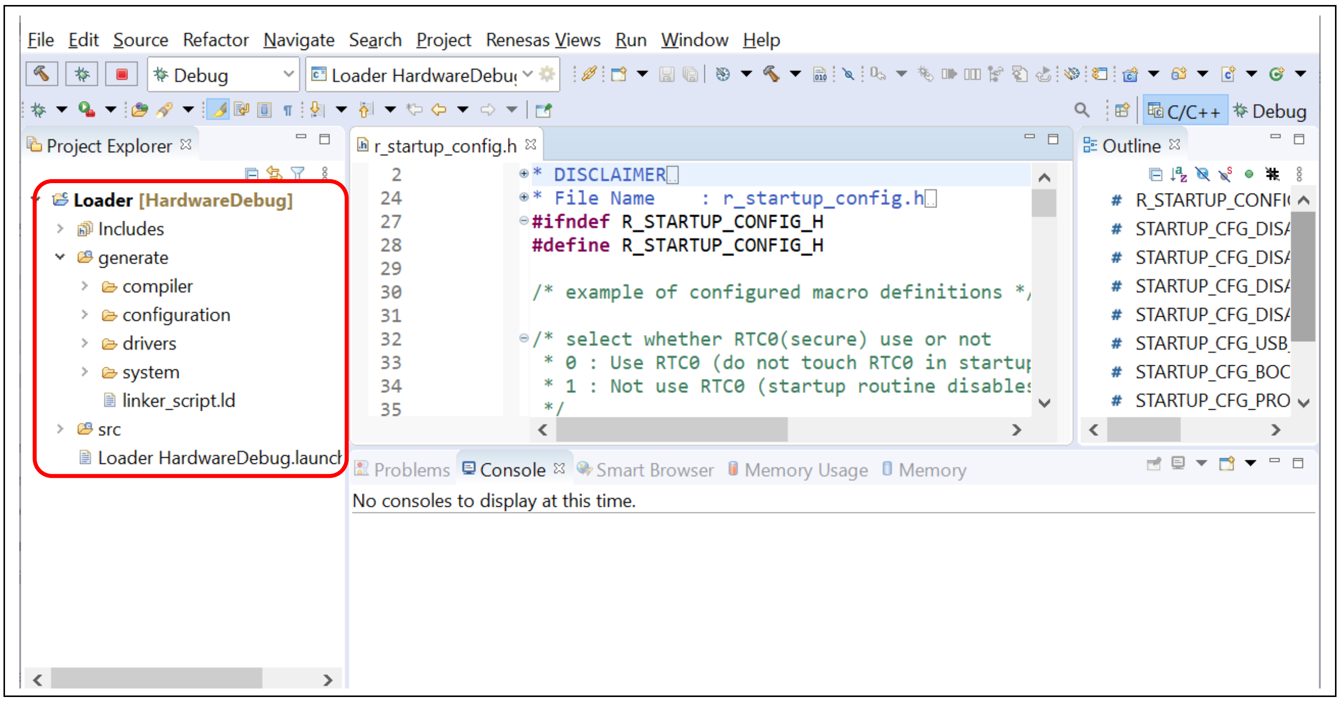

New GCC for Renesas RZ Executable Project

Summary of project "Loader"

```
Device: R7S921053
Toolchain Version: 6.3.1.20170620
Files added:
    generate/system/
    generate/compiler/
    generate/drivers/
    generate/system/
    generate/linker_script.ld
```

Figure 3-14. New Project Creation Wizard (Summary)

10. A new C project named 'Loader' will be created.



```
 * DISCLAIMER
 * File Name    : r_startup_config.h
#ifndef R_STARTUP_CONFIG_H
#define R_STARTUP_CONFIG_H

/* example of configured macro definitions */

/* select whether RTC0(secure) use or not
 * 0 : Use RTC0 (do not touch RTC0 in startup
 * 1 : Not use RTC0 (startup routine disable
 */
```

```
No consoles to display at this time.
```

Figure 3-15. New C Project Created

11. Next, create an Application project.

Click **File → New → C/C++ Project** to open the **New Project Creation Wizard**.

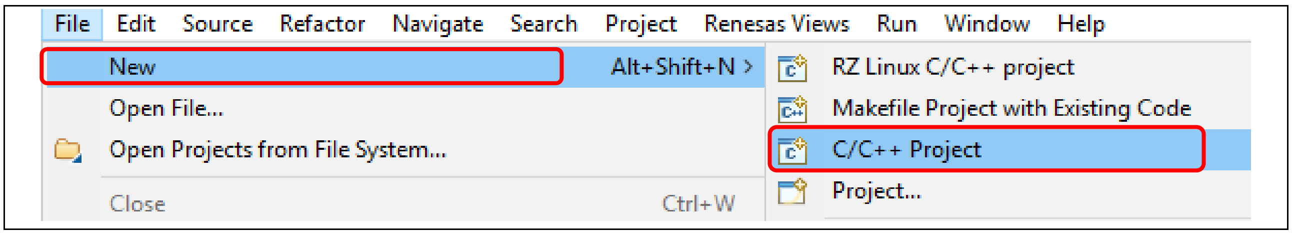
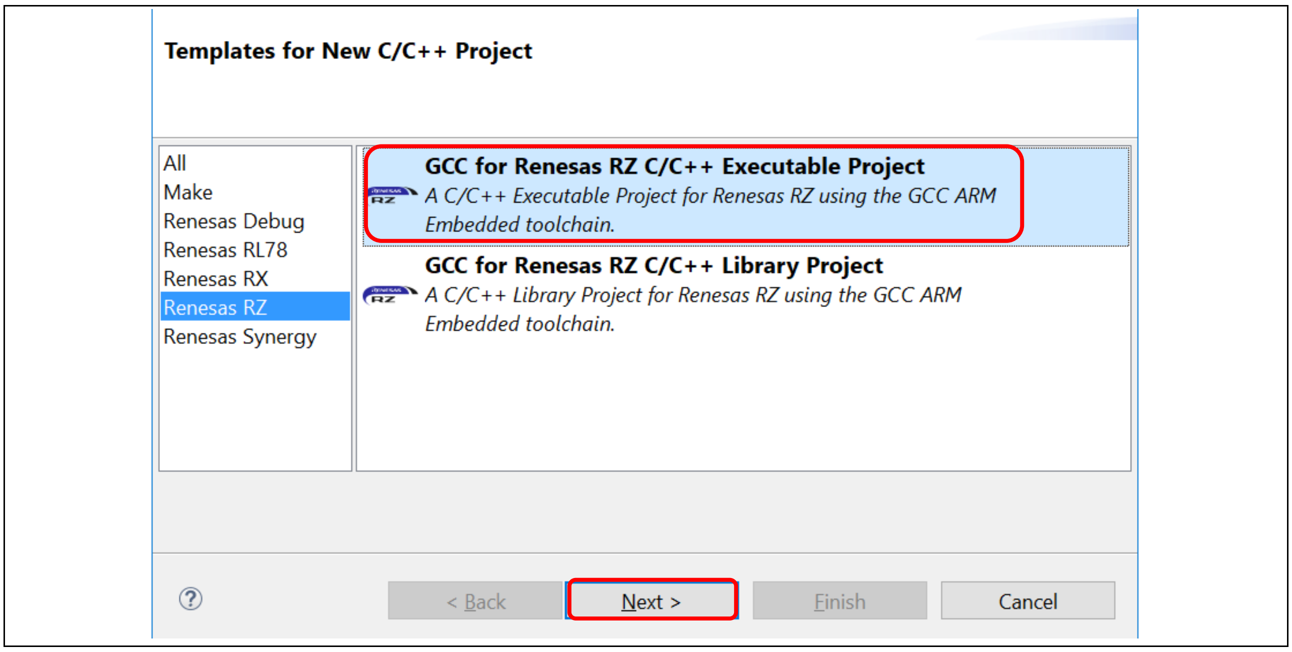

File Edit Source Refactor Navigate Search Project Renesas Views Run Window Help
New Alt+Shift+N >
Open File...
Open Projects from File System...
Close Ctrl+W
RZ Linux C/C++ project
Makefile Project with Existing Code
C/C++ Project
Project...

Figure 3-16 Open New Project Creation Wizard

12. Select the template for the new project (for example, Renesas RZ: 'GCC for Renesas RZ C/C++ Executable Project'). Click **Next** to proceed.



Templates for New C/C++ Project

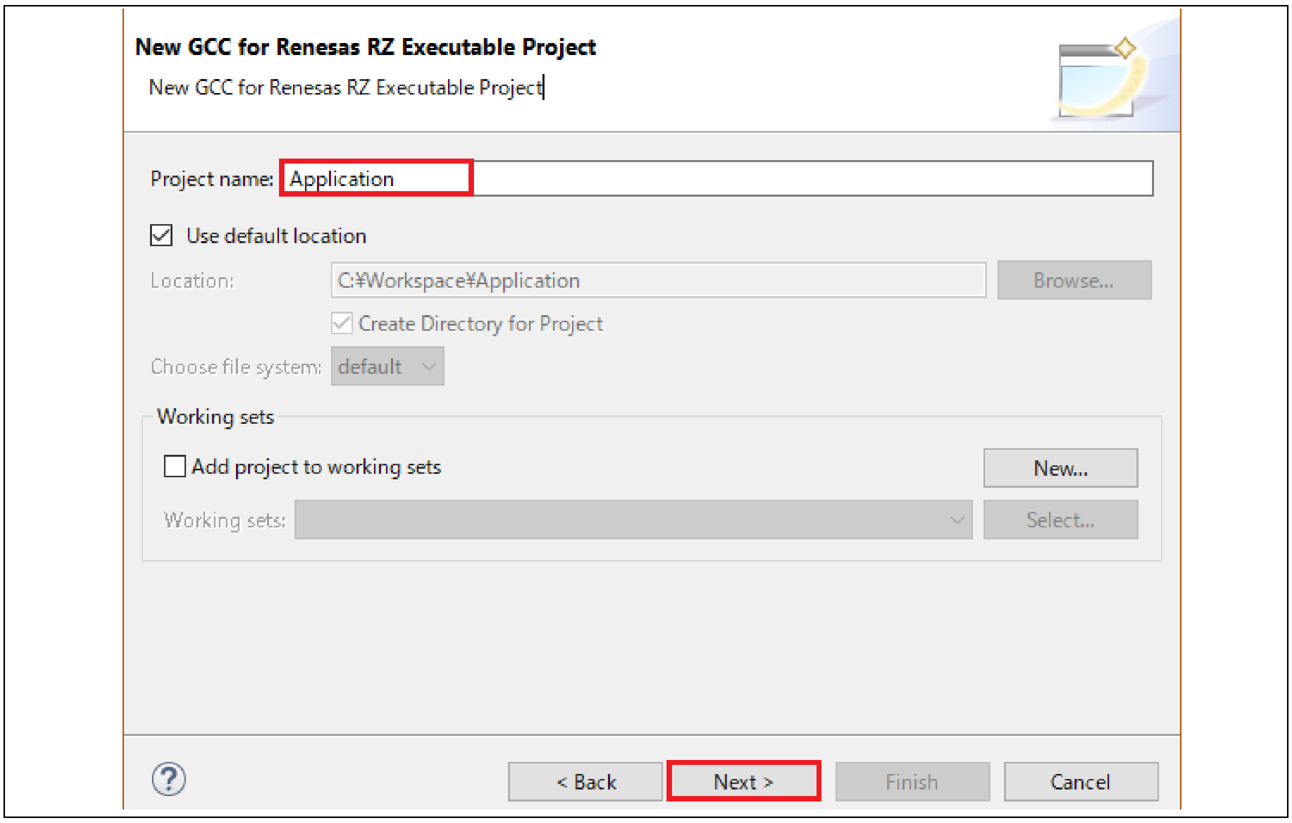
All
Make
Renesas Debug
Renesas RL78
Renesas RX
Renesas RZ
Renesas Synergy

GCC for Renesas RZ C/C++ Executable Project
A C/C++ Executable Project for Renesas RZ using the GCC ARM Embedded toolchain.

GCC for Renesas RZ C/C++ Library Project
A C/C++ Library Project for Renesas RZ using the GCC ARM Embedded toolchain.

Figure 3-17. Select Project Template

13. Now, create the Application Project. Enter the project name ('Application'). Click **Next** to proceed.



New GCC for Renesas RZ Executable Project
New GCC for Renesas RZ Executable Project

Project name: Application

☑ Use default location

Location: C:¥Workspace¥Application

☑ Create Directory for Project

Choose file system: default

Working sets

☐ Add project to working sets

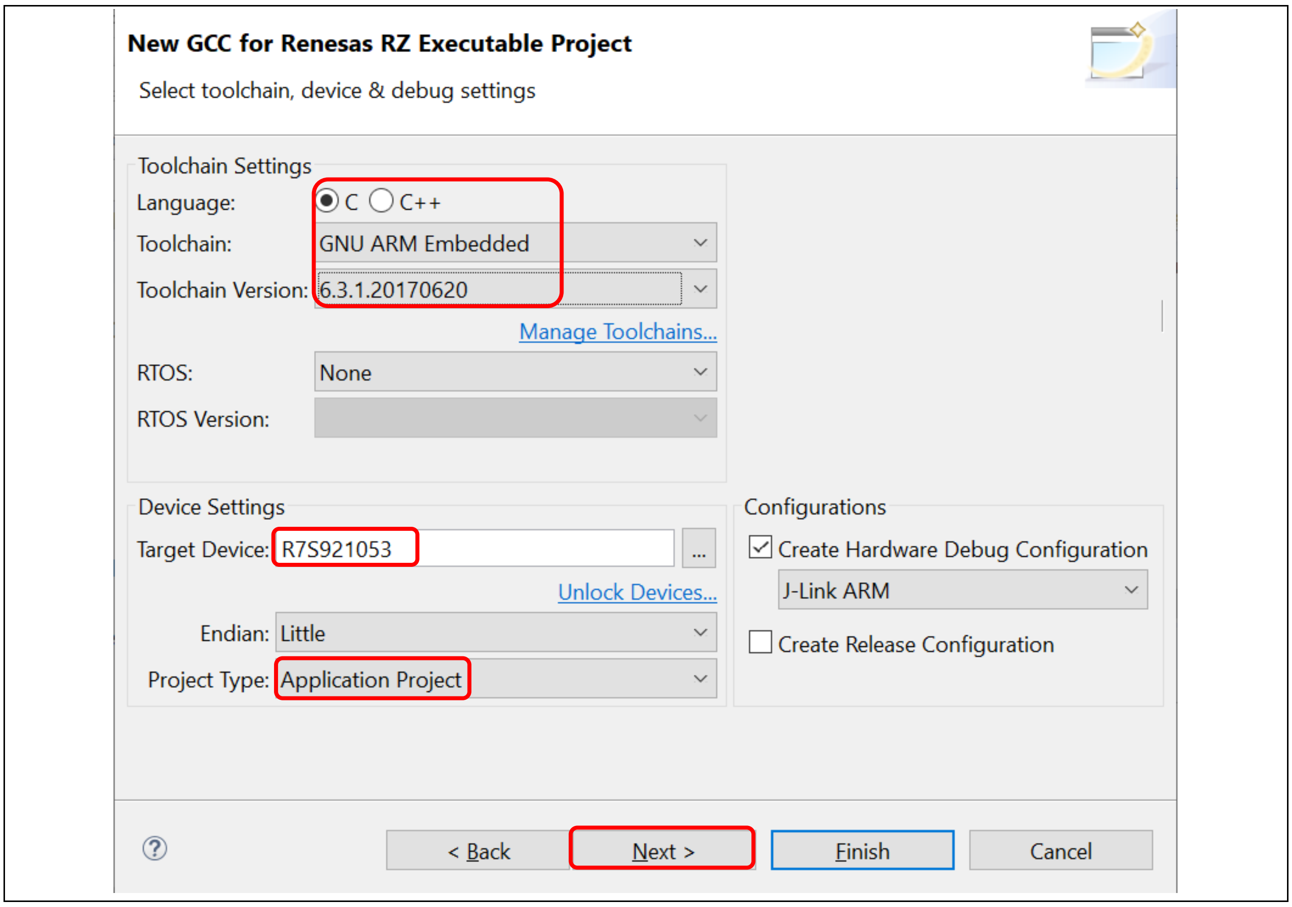
Working sets:

Figure 3-18. Specify Project Name

14. Select the following settings.

- Language: 'C' or 'C++'
- Toolchain: 'GCC ARM Embedded'
- Toolchain Version: '6.3.1.20170620'
- Target Device: 'R7S921053'
- Project Types: 'Application Project'

Select **Create Hardware Debug Configuration** and then **J-Link ARM**. Click **Next** to proceed.



New GCC for Renesas RZ Executable Project

Select toolchain, device & debug settings

Toolchain Settings

Language: [✓ C] C++

Toolchain: GNU ARM Embedded

Toolchain Version: 6.3.1.20170620

Manage Toolchains...

RTOS: None

RTOS Version:

Device Settings

Target Device: R7S921053

Unlock Devices...

Endian: Little

Project Type: Application Project

Configurations

[✓] Create Hardware Debug Configuration

J-Link ARM

[ ] Create Release Configuration

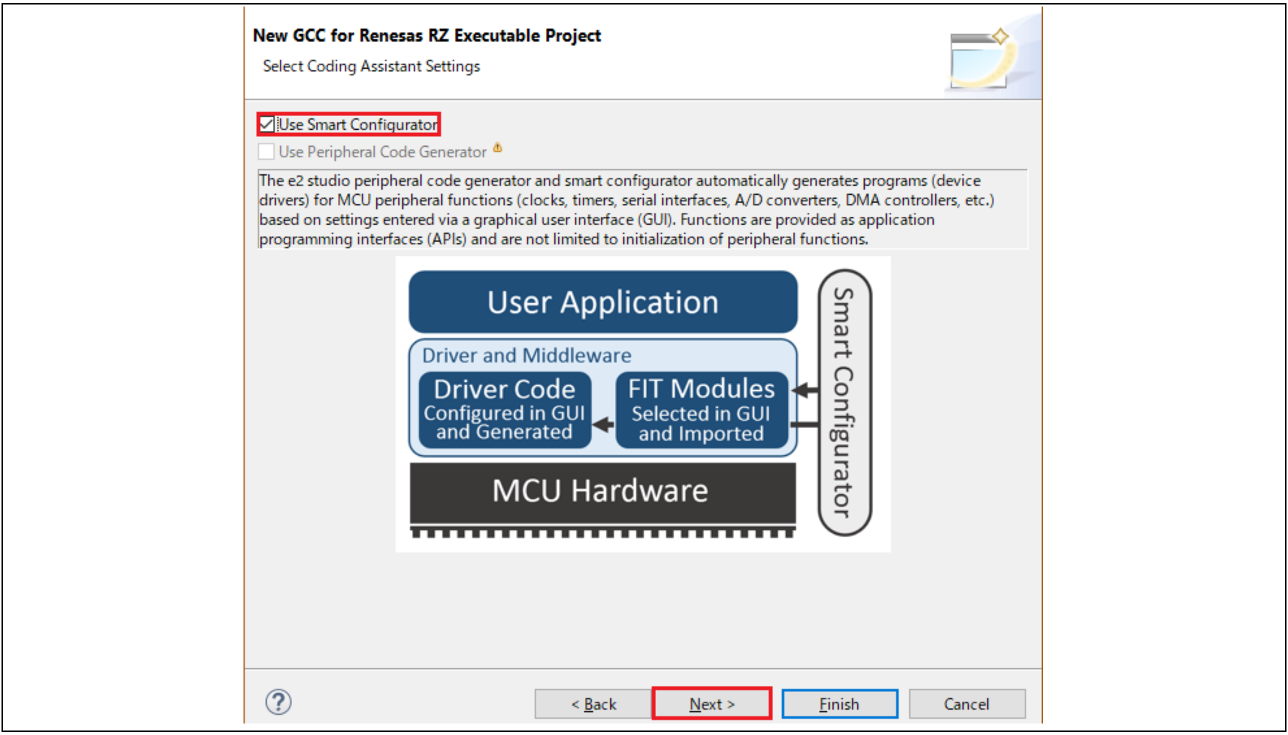
Figure 3-19. Select Toolchain, Device & Debug Settings: e.g. RZ/A2M Renesas Starter Kit

15. The Coding Assistant feature can be selected if required.

Note:

- *Smart Configurator* supports a single user interface that combines the functionalities of Code Generator and FIT Configurator. Smart Configurator encompasses a unified clock configuration view, an interrupt configuration view and a pin configuration view.

Select **Use Smart Configurator**, click **Next** to proceed.



New GCC for Renesas RZ Executable Project

Select Coding Assistant Settings

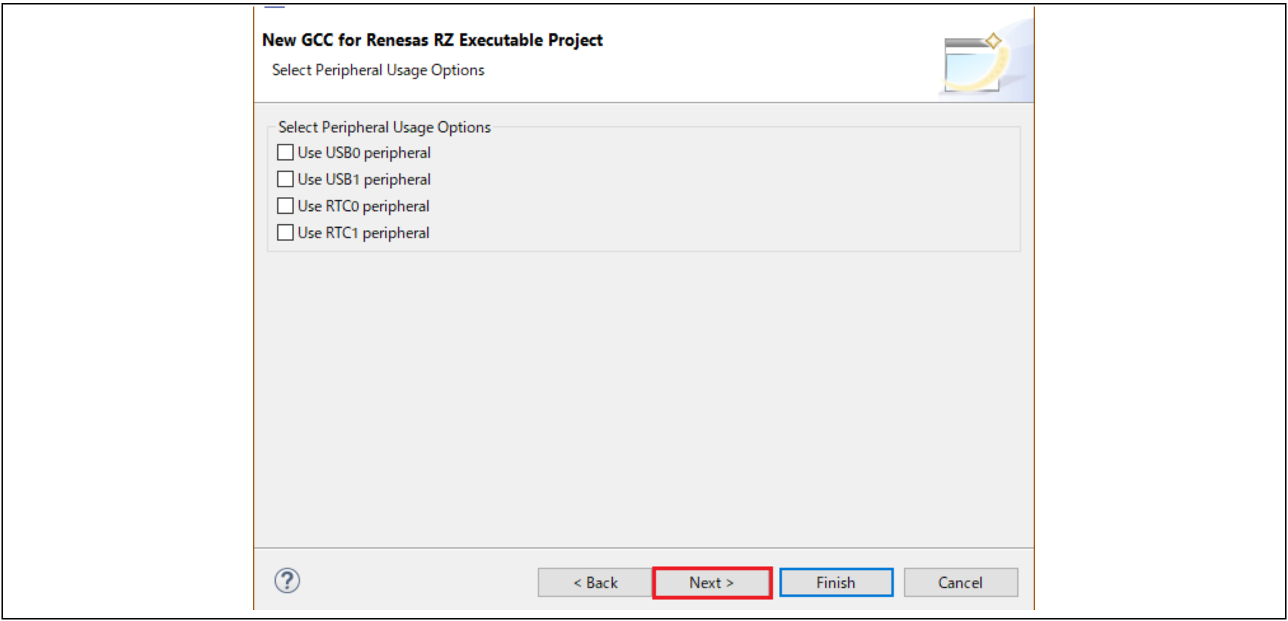
☑ Use Smart Configurator

☐ Use Peripheral Code Generator

The e2 studio peripheral code generator and smart configurator automatically generates programs (device drivers) for MCU peripheral functions (clocks, timers, serial interfaces, A/D converters, DMA controllers, etc.) based on settings entered via a graphical user interface (GUI). Functions are provided as application programming interfaces (APIs) and are not limited to initialization of peripheral functions.

Figure 3-20. Select Coding Assistant Settings

16. In the initial settings, USB and RTC are not used. When using USB and RTC, select the peripheral module / channel to be used.



New GCC for Renesas RZ Executable Project

Select Peripheral Usage Options

Select Peripheral Usage Options

☐ Use USB0 peripheral

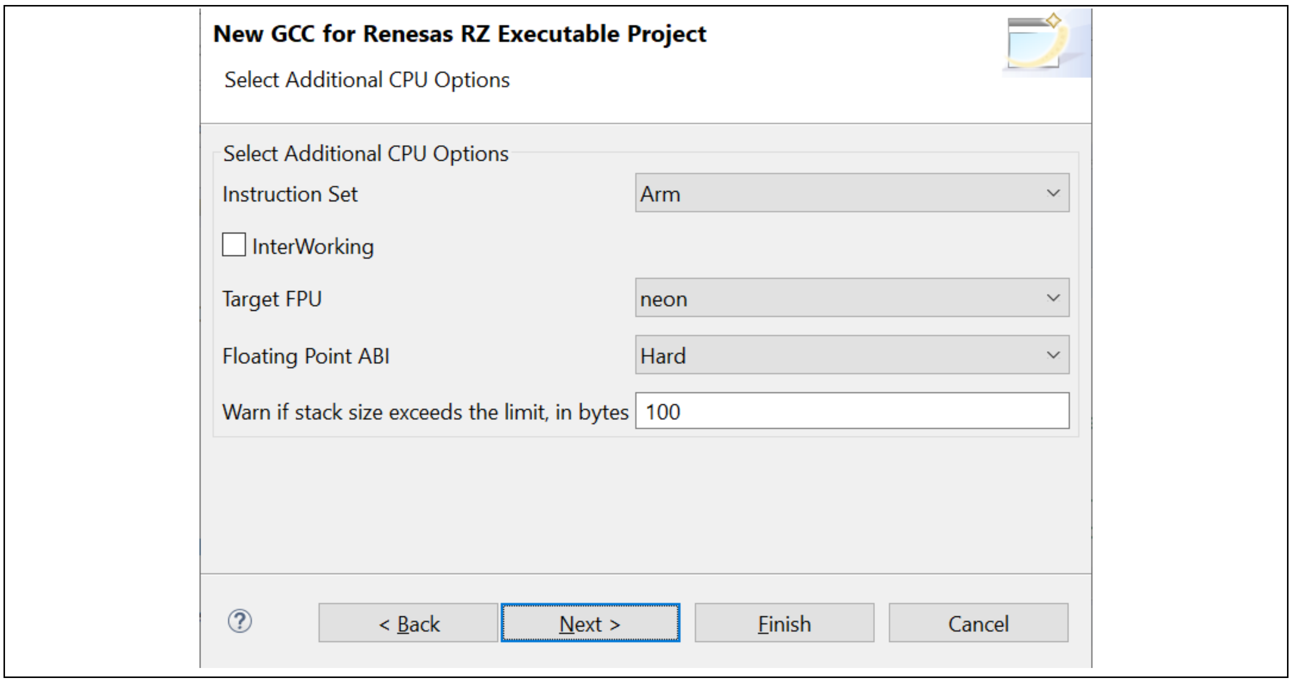
☐ Use USB1 peripheral

☐ Use RTC0 peripheral

☐ Use RTC1 peripheral

Figure 3-21. Select Peripheral Usage Options

17. Keep the default settings for additional CPU options and click **Next** to proceed.



New GCC for Renesas RZ Executable Project

Select Additional CPU Options

Select Additional CPU Options

Instruction Set: Arm

☐ InterWorking

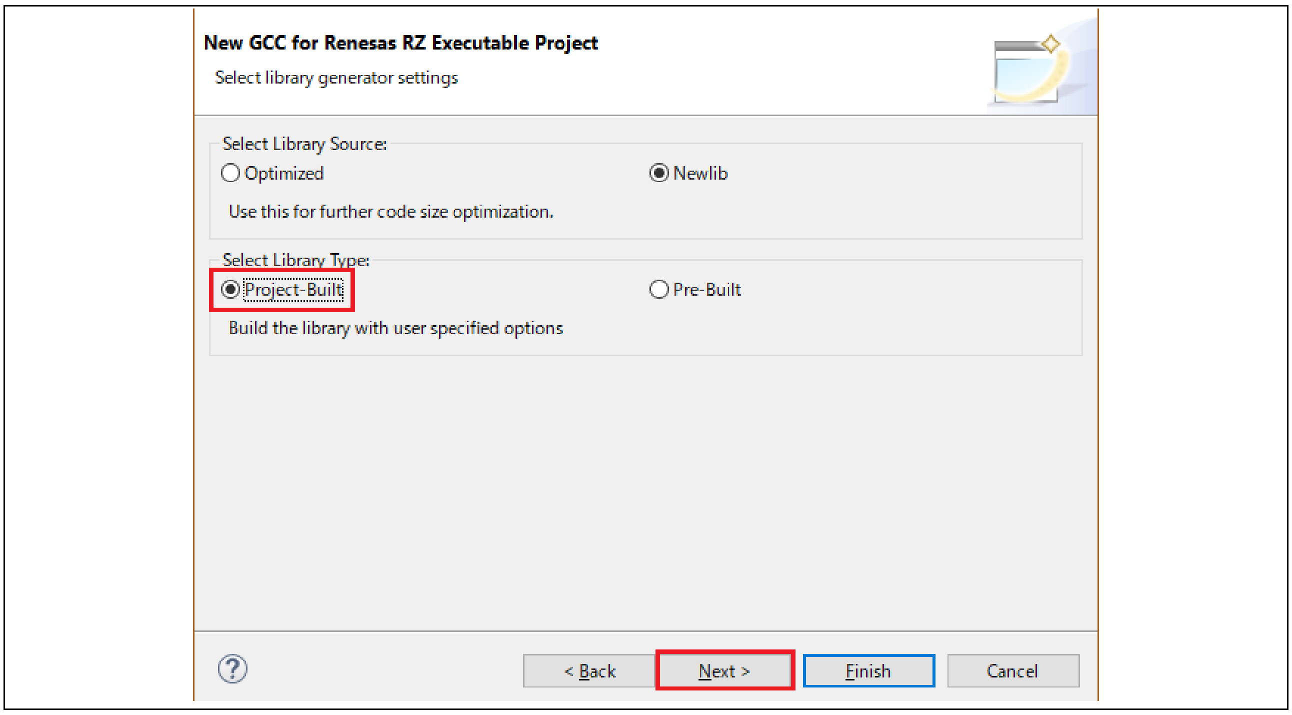
Target FPU: neon

Floating Point ABI: Hard

Warn if stack size exceeds the limit, in bytes: 100

Figure 3-22. Select Additional CPU Options

18. Select **Project-Built** and click **Next** to proceed.



New GCC for Renesas RZ Executable Project

Select library generator settings

Select Library Source:

○ Optimized ● Newlib

Use this for further code size optimization.

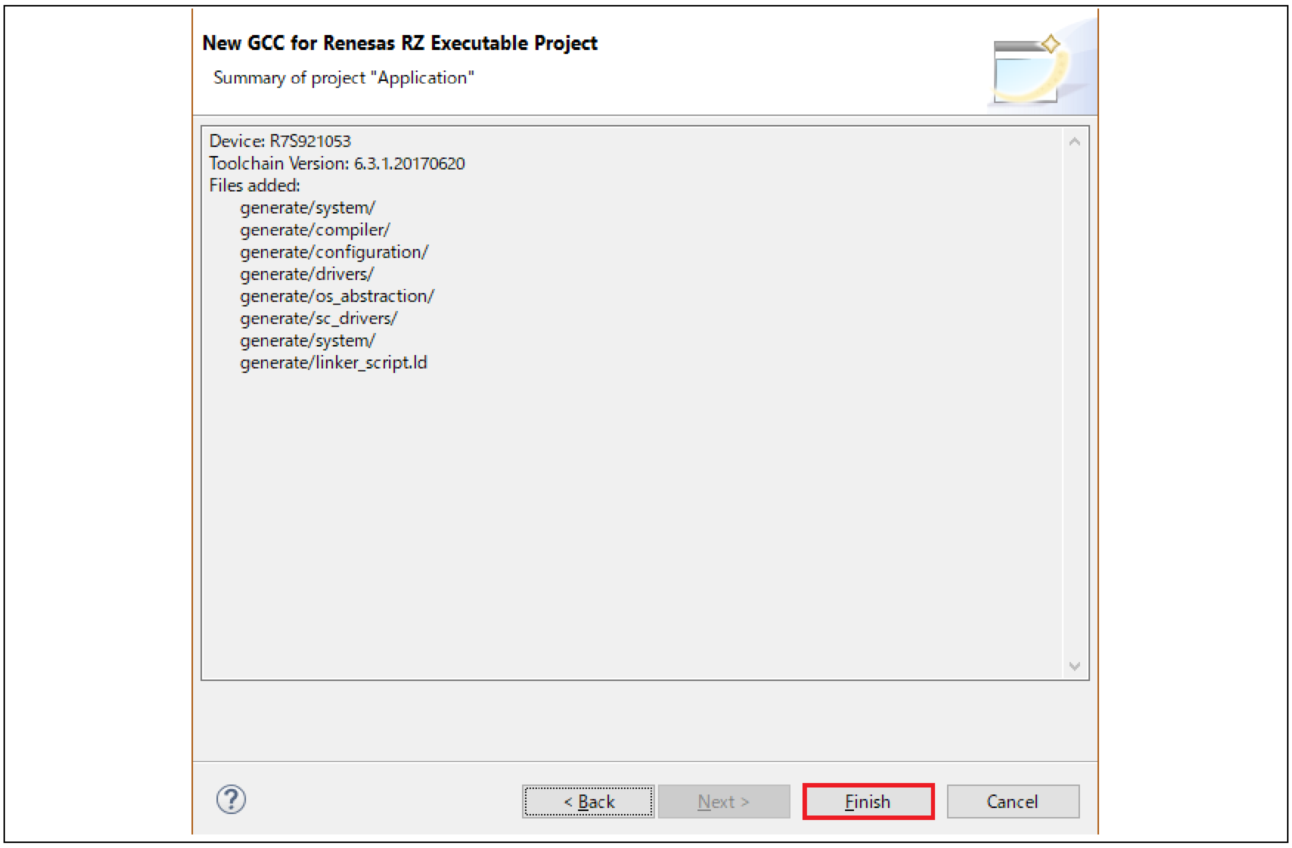
Select Library Type:

● Project-Built ○ Pre-Built

Build the library with user specified options

Figure 3-23. Select Library Generator Settings

19. A project summary is displayed. Click **Finish** to generate the project.

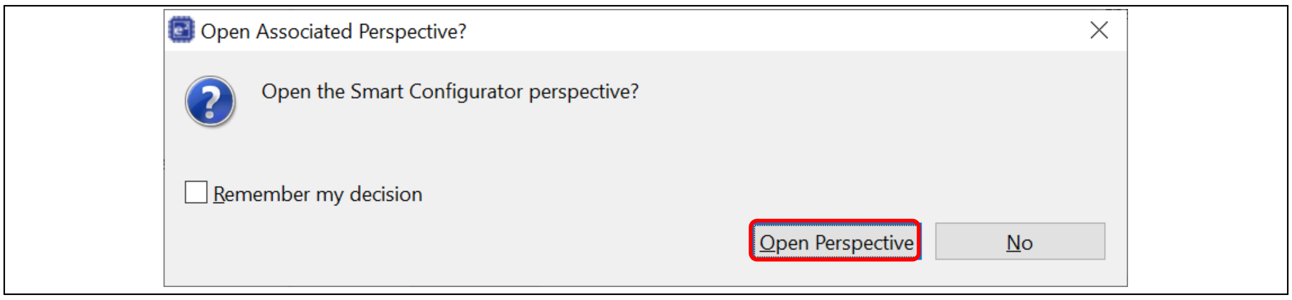

New GCC for Renesas RZ Executable Project

Summary of project "Application"

Device: R7S921053
Toolchain Version: 6.3.1.20170620
Files added:
generate/system/
generate/compiler/
generate/configuration/
generate/drivers/
generate/os_abstraction/
generate/sc_drivers/
generate/system/
generate/linker_script.ld

Figure 3-24. New Project Creation Wizard (Summary)

20. Click **Open Perspective**.


Open Associated Perspective?

Open the Smart Configurator perspective?

Remember my decision

Figure 3-25. Open Associated Perspective Wizard

21. A new C project named 'Application' is created.

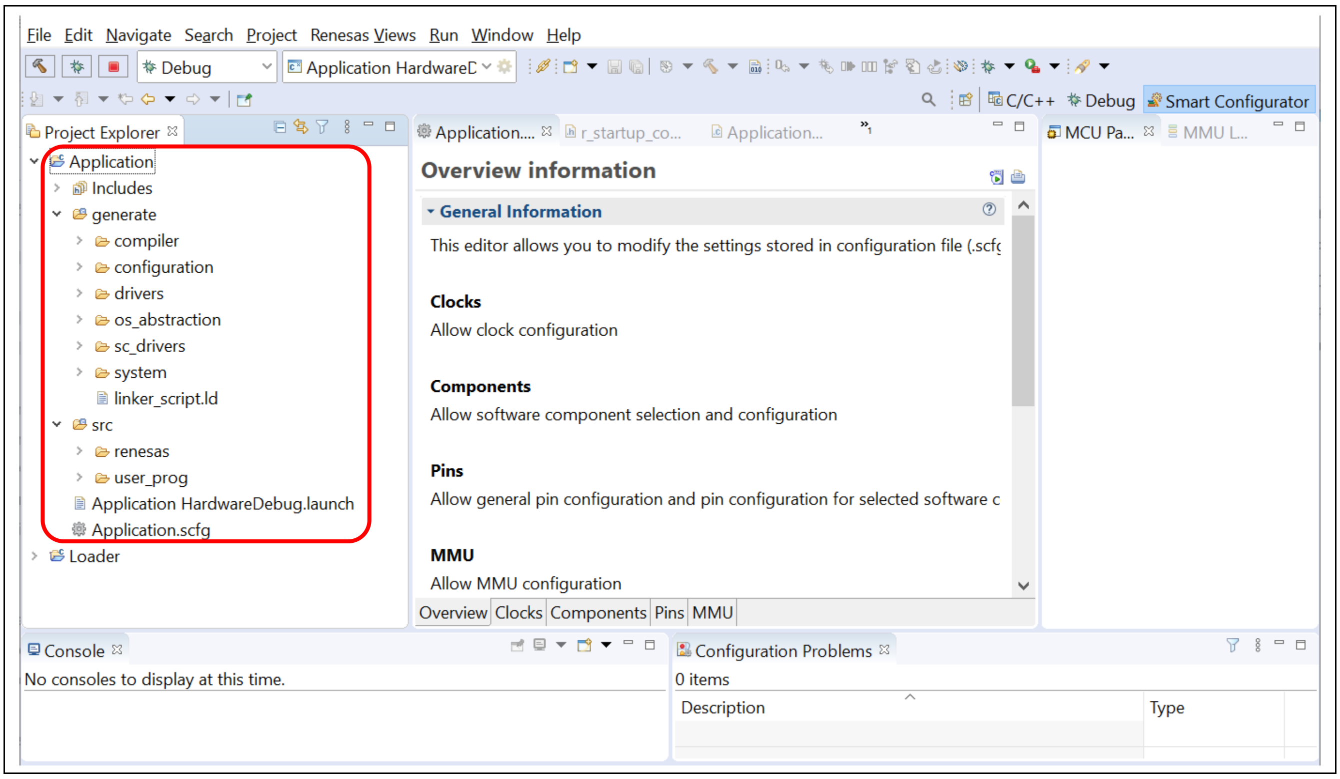


Figure 3-26. New C Project Created

This project consists of an application file `Application.c` and a set of automatically generated files. All these project and source files are listed in the **Project Explorer** panel reflecting the folder structure of the project, just as seen in the standard file explorer.

Notes for backing up projects:

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

3.2.2 Project Generation for RZ groups (RZ/A1, RZ/T)

This chapter illustrates an example when using the RZ/A1H Renesas Starter Kit.

The created project runs within the large-capacity internal RAM of the RZ/A1. When executing with flash memory, change the program according to the specifications of the serial flash memory and memory placement.

RZ/T project generation is almost same with RZ/A1H.

For *RZ/G*, e^2 studio supports **RZ Linux C/C++ project** for developing Linux application. The **Support for RZ/G** option has to be selected during e^2 studio installation. Refer to **Figure 2-2**.

To create a new project, invoke e^2 studio IDE from the Windows Start menu and specify a workspace directory.

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

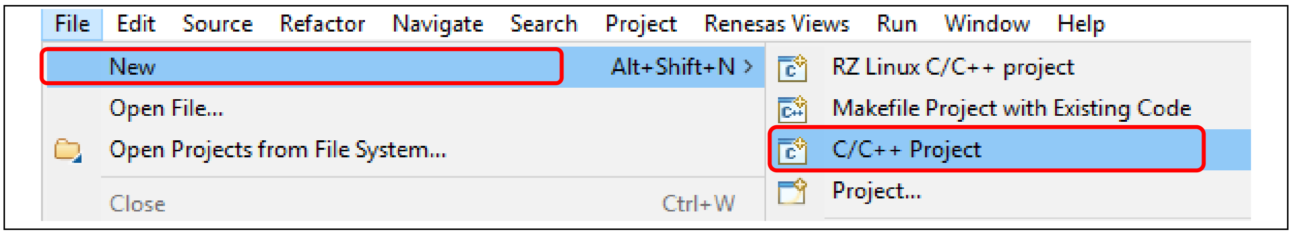
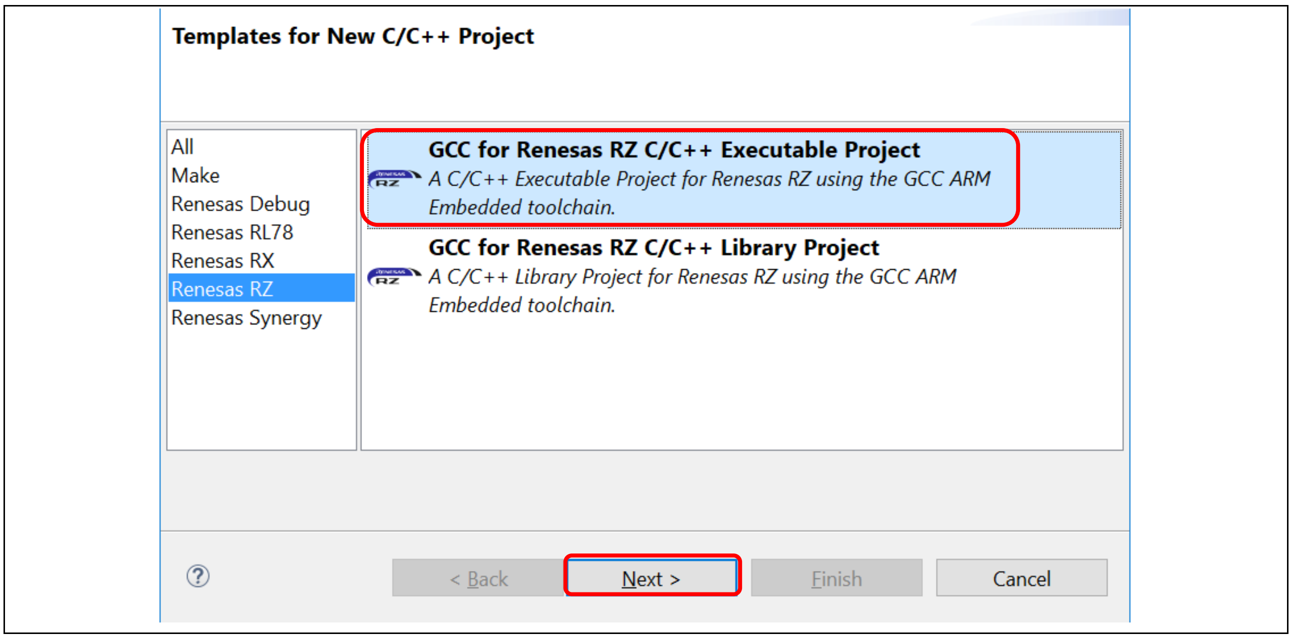


Figure 3-27. Open New Project Creation Wizard

2. Select template for the new project (for example, Renesas RZ: 'GCC for Renesas RZ C/C++ Executable Project'). Click **Next** to proceed.

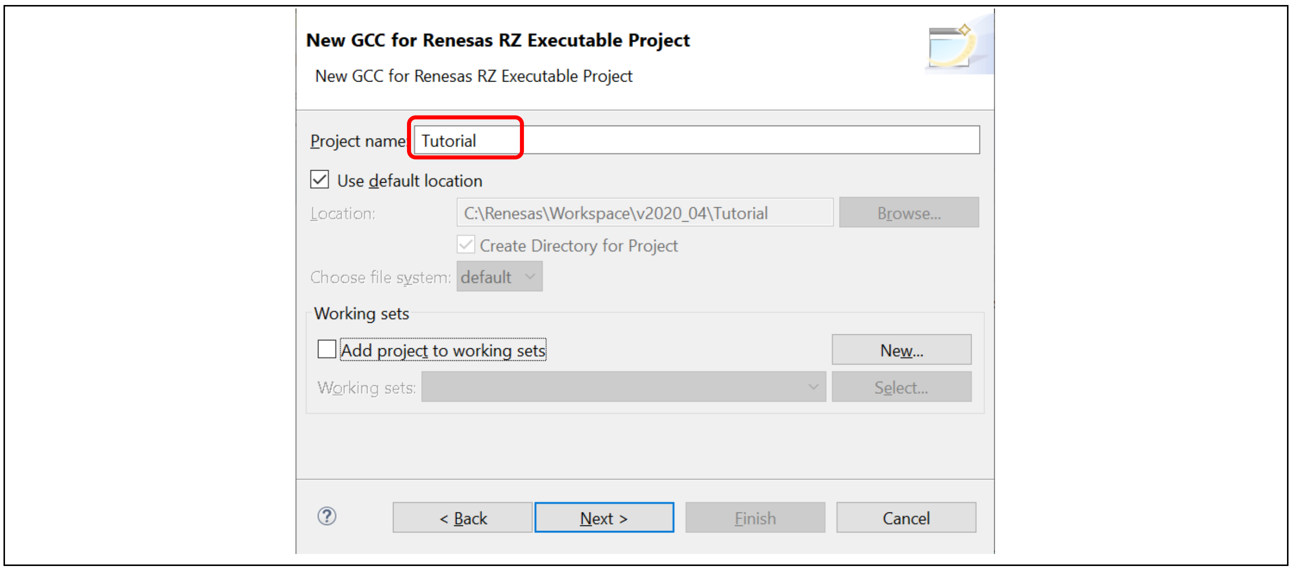


GCC for Renesas RZ C/C++ Executable Project
A C/C++ Executable Project for Renesas RZ using the GCC ARM Embedded toolchain.

GCC for Renesas RZ C/C++ Library Project
A C/C++ Library Project for Renesas RZ using the GCC ARM Embedded toolchain.

Figure 3-28. Select Project Template

3. Enter the project name. Click **Next** to proceed.



New GCC for Renesas RZ Executable Project

New GCC for Renesas RZ Executable Project

Project name: Tutorial

☑ Use default location

Location: C:\Renesas\Workspace\v2020_04\Tutorial

☑ Create Directory for Project

Choose file system: default

Working sets

☐ Add project to working sets

Working sets:

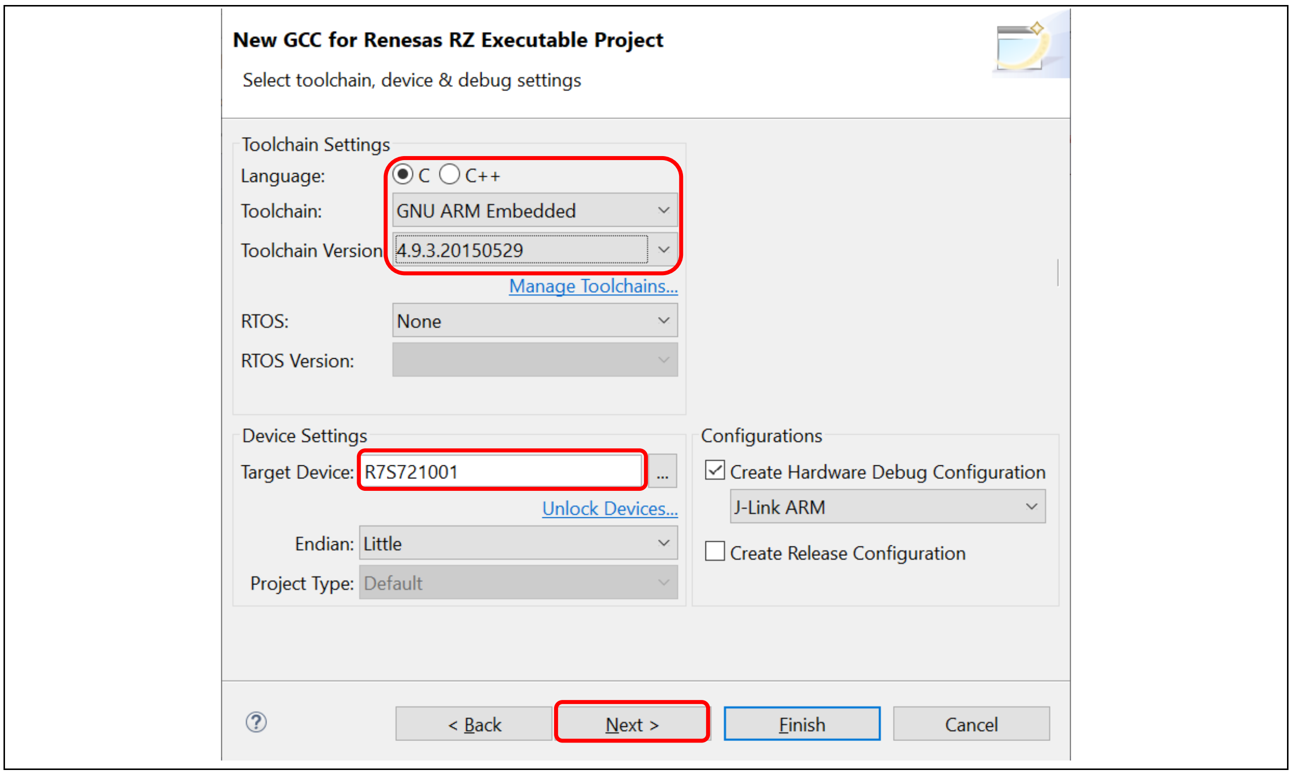
Figure 3-29. Specify Project Name

Select the following settings.

For example using the RZ/A1H Renesas Starter Kit:

- Language: 'C'
- Toolchain: 'GCC ARM Embedded'
- Toolchain Version: '4.9.20150529'
- Target Device: 'R7S721001'

Select **Create Hardware Debug Configuration** and then **J-Link ARM**. Click **Next** to proceed.



New GCC for Renesas RZ Executable Project

Select toolchain, device & debug settings

Toolchain Settings

Language: [✓ C] C++

Toolchain: GNU ARM Embedded

Toolchain Version: 4.9.3.20150529

RTOS: None

RTOS Version:

Device Settings

Target Device: R7S721001

Endian: Little

Project Type: Default

Configurations

☑ Create Hardware Debug Configuration

J-Link ARM

☐ Create Release Configuration

Figure 3-30. Select toolchain, device & debug settings: e.g. RZ/A1H Renesas Starter Kit

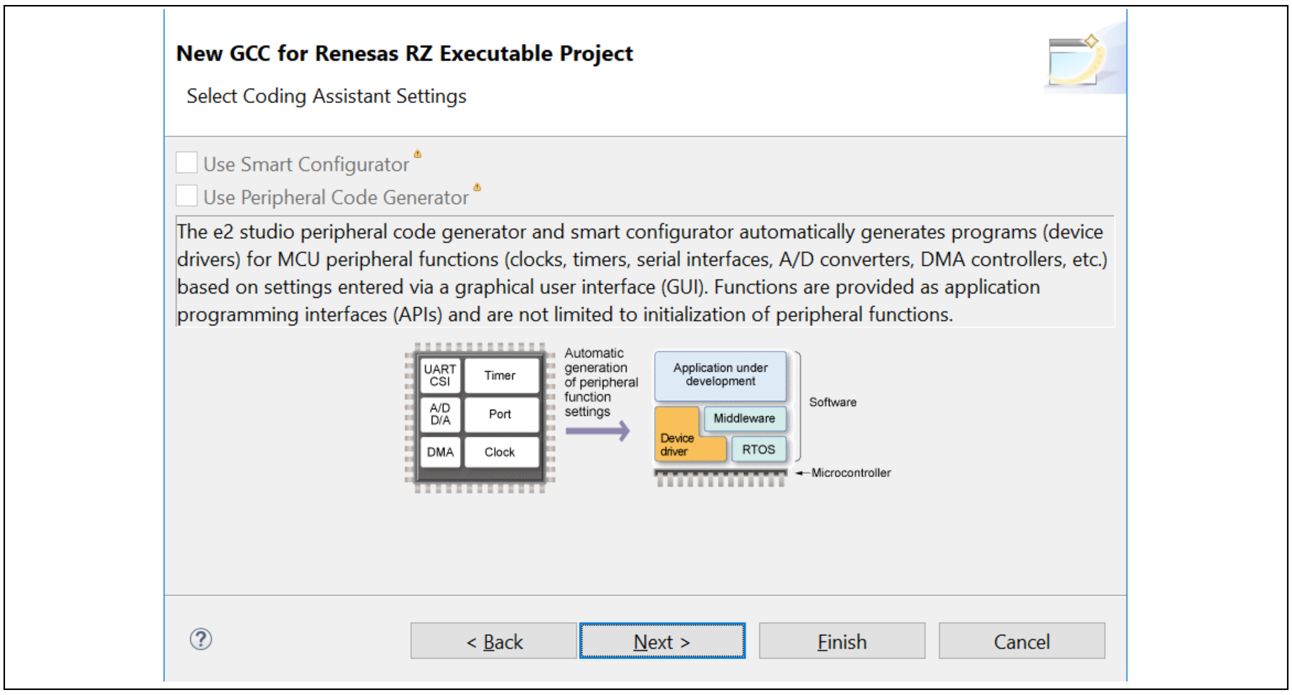
4. The Coding Assistant feature can be selected if required.

Note:

- The *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 peripheral initialization.
- *Smart Configurator* supports a single user interface that combines the functionalities of Code Generator and Middleware Configurator. Smart Configurator encompasses a unified clock configuration view, an interrupt configuration view and a pin configuration view.

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

6. Click **Next** to proceed.



New GCC for Renesas RZ Executable Project

Select Coding Assistant Settings

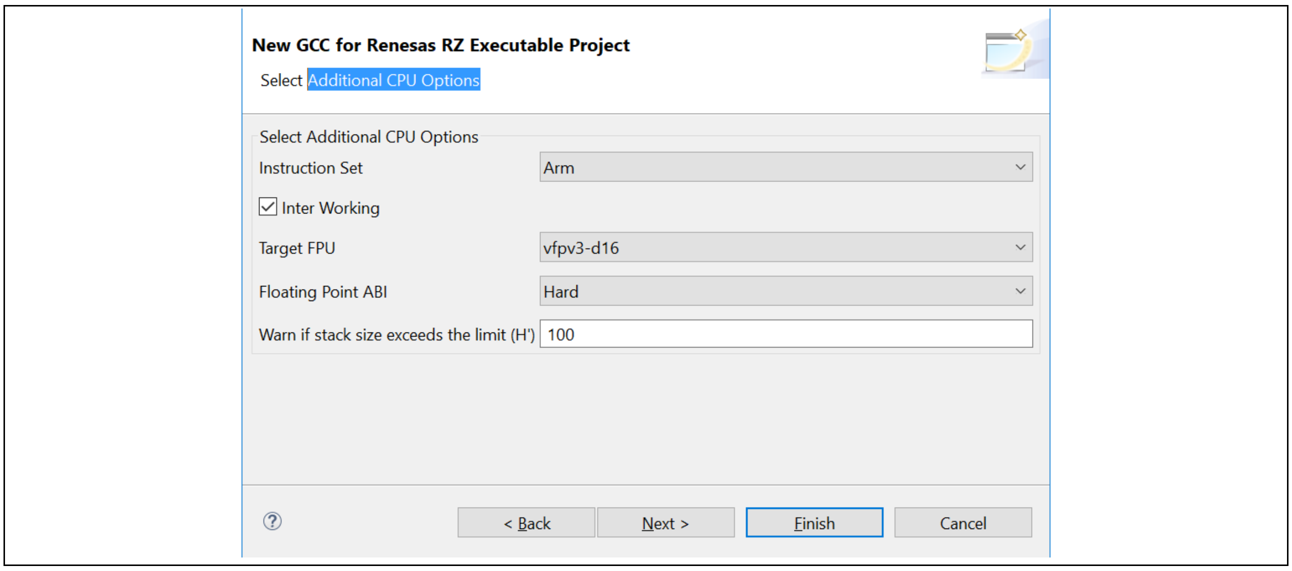
☐ Use Smart Configurator

☐ Use Peripheral Code Generator

The e2 studio peripheral code generator and smart configurator automatically generates programs (device drivers) for MCU peripheral functions (clocks, timers, serial interfaces, A/D converters, DMA controllers, etc.) based on settings entered via a graphical user interface (GUI). Functions are provided as application programming interfaces (APIs) and are not limited to initialization of peripheral functions.

Figure 3-31. Select Coding Assistant Settings

7. Keep the default settings for additional CPU options and click **Next** to proceed.



New GCC for Renesas RZ Executable Project

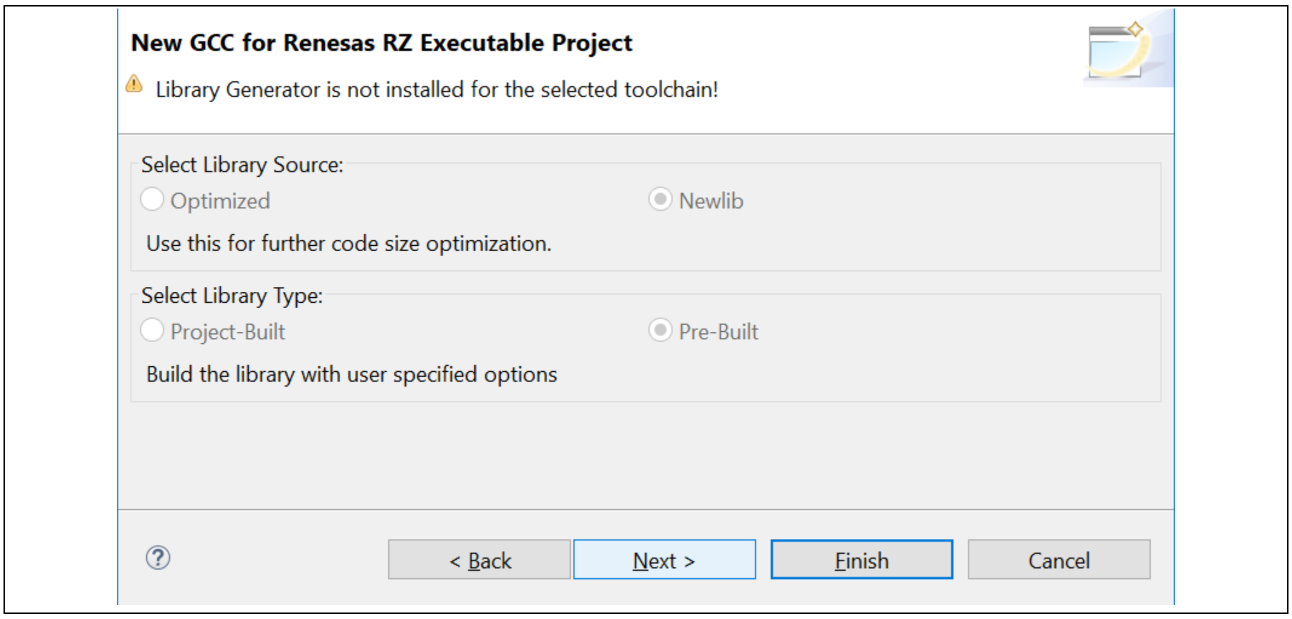
Select Additional CPU Options

Select Additional CPU Options

Instruction Set	Arm
☑ Inter Working	
Target FPU	vfpv3-d16
Floating Point ABI	Hard
Warn if stack size exceeds the limit (H')	100

Figure 3-32. Select Additional CPU Options

8. If the selected compiler was not updated by **Libgen Update for GNU ARM Embedded Toolchains**, a warning message about the Library Generator will be shown in the next dialog box. Without the Library Generator, there may be project build errors during compilation.
Refer to chapter 2.2.2 and run **Libgen Update for GNU ARM Embedded Toolchains**.


New GCC for Renesas RZ Executable Project

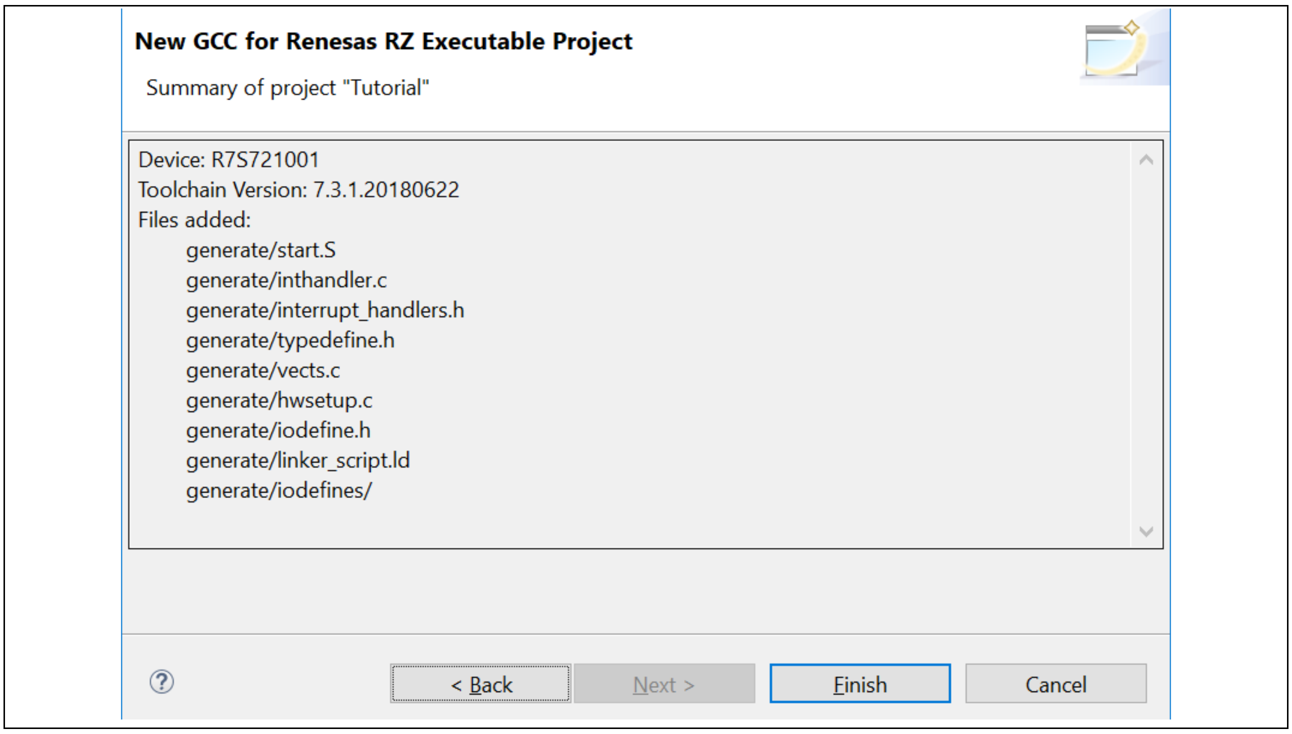
Library Generator is not installed for the selected toolchain!

Select Library Source:
Optimized [✓ Newlib]
Use this for further code size optimization.

Select Library Type:
Project-Built [✓ Pre-Built]
Build the library with user specified options

Figure 3-33. Library Generator Settings Warning Message

9. A project summary is displayed. Click **Finish** to generate the project.

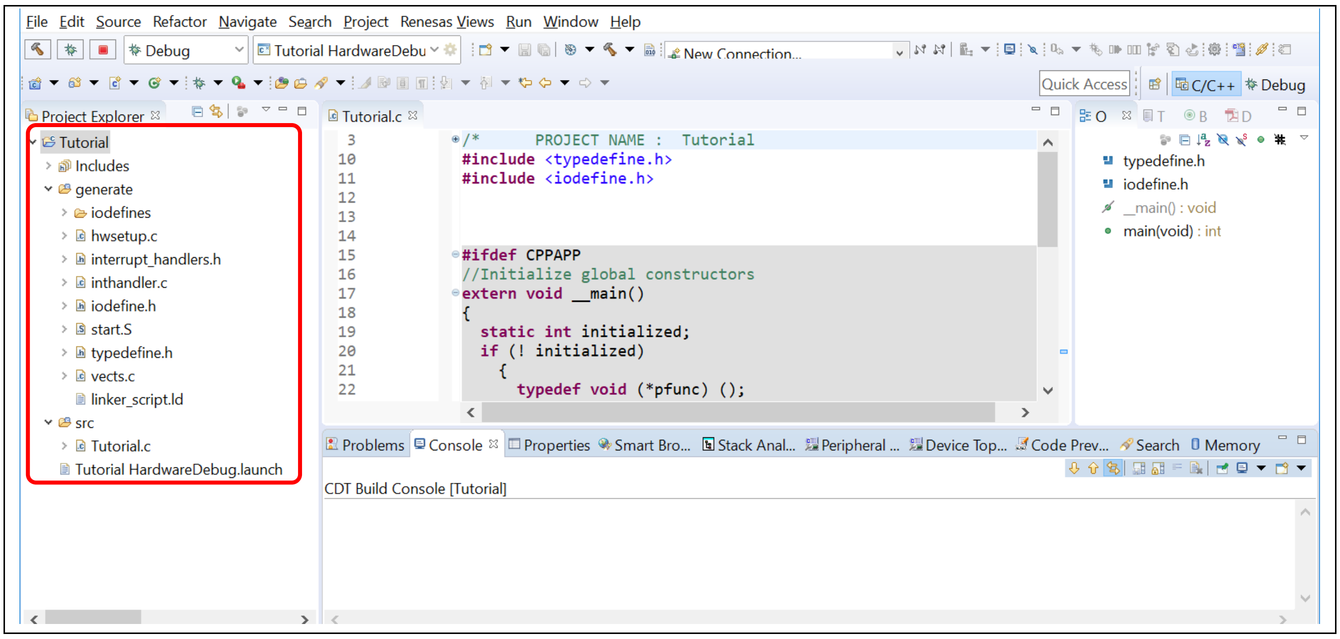

New GCC for Renesas RZ Executable Project

Summary of project "Tutorial"

```
Device: R7S721001
Toolchain Version: 7.3.1.20180622
Files added:
      generate/start.S
      generate/inthandler.c
      generate/interrupt_handlers.h
      generate/typedefine.h
      generate/vects.c
      generate/hwsetup.c
      generate/iodefine.h
      generate/linker_script.ld
      generate/iodefines/
```

Figure 3-34. New Project Creation Wizard (Summary)

10. A new C project named 'Tutorial' will be created.



```
/*      PROJECT NAME :  Tutorial
#include <typedefine.h>
#include <iodefine.h>



#ifdef CPPAPP
//Initialize global constructors
extern void __main()
{
  static int initialized;
  if (! initialized)
    {
      typedef void (*pfunc) ();
```

Figure 3-35. New C Project Created

This project consists of an application file `Tutorial.c` and a set of automatically generated files. All these project and source files are listed in the **Project Explorer** panel reflecting the folder structure of the project, just as seen in the standard file explorer.

Notes for backing up projects:

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

4. Build

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

4.1 Build Option Settings

A new project generated with default options should run well, but it is possible to change build options with the following configuration dialogs.

1. Right click on project 'Tutorial' and select **Properties** or use shortcut keys **Alt** + **Enter** to open the **Properties** window.

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

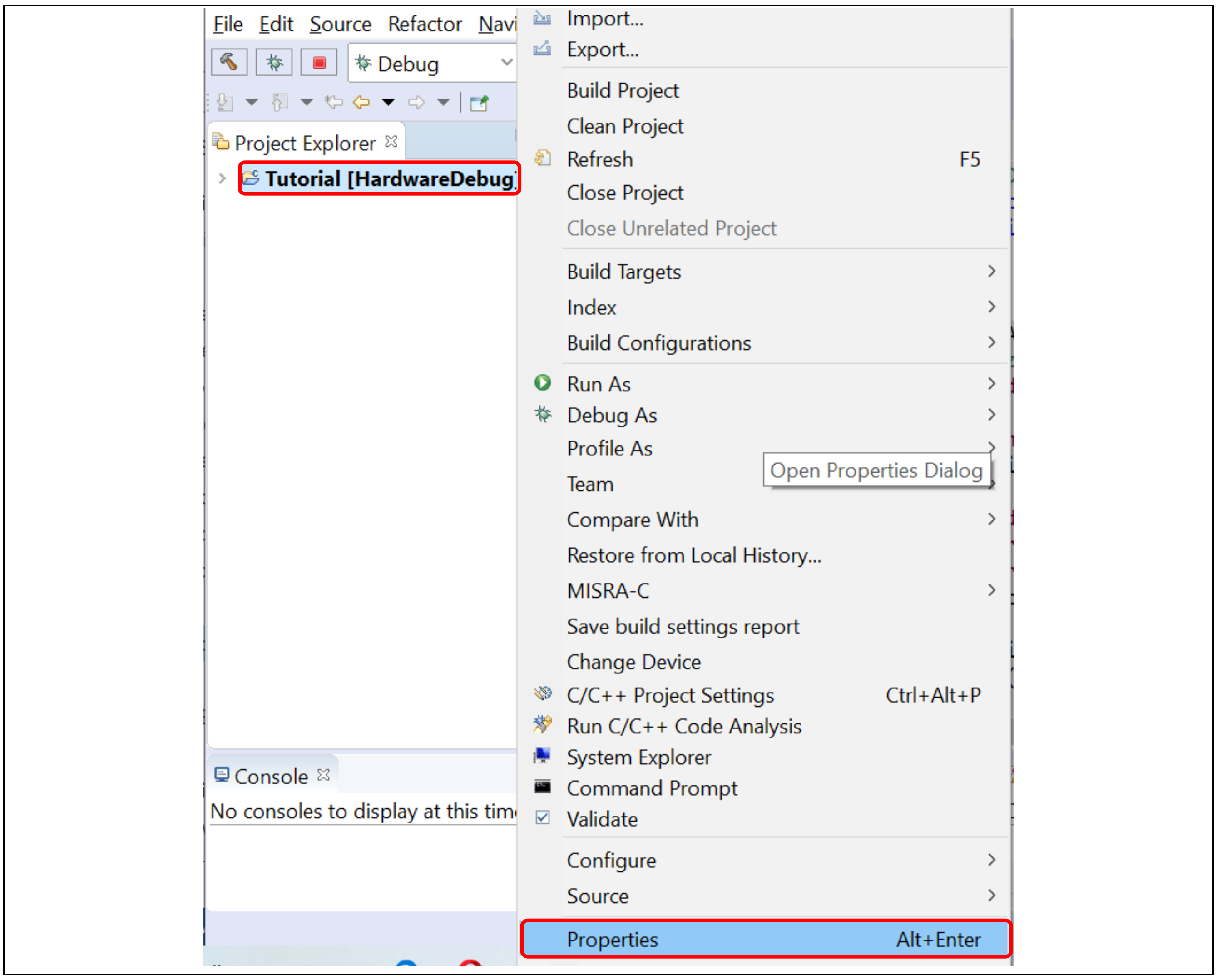
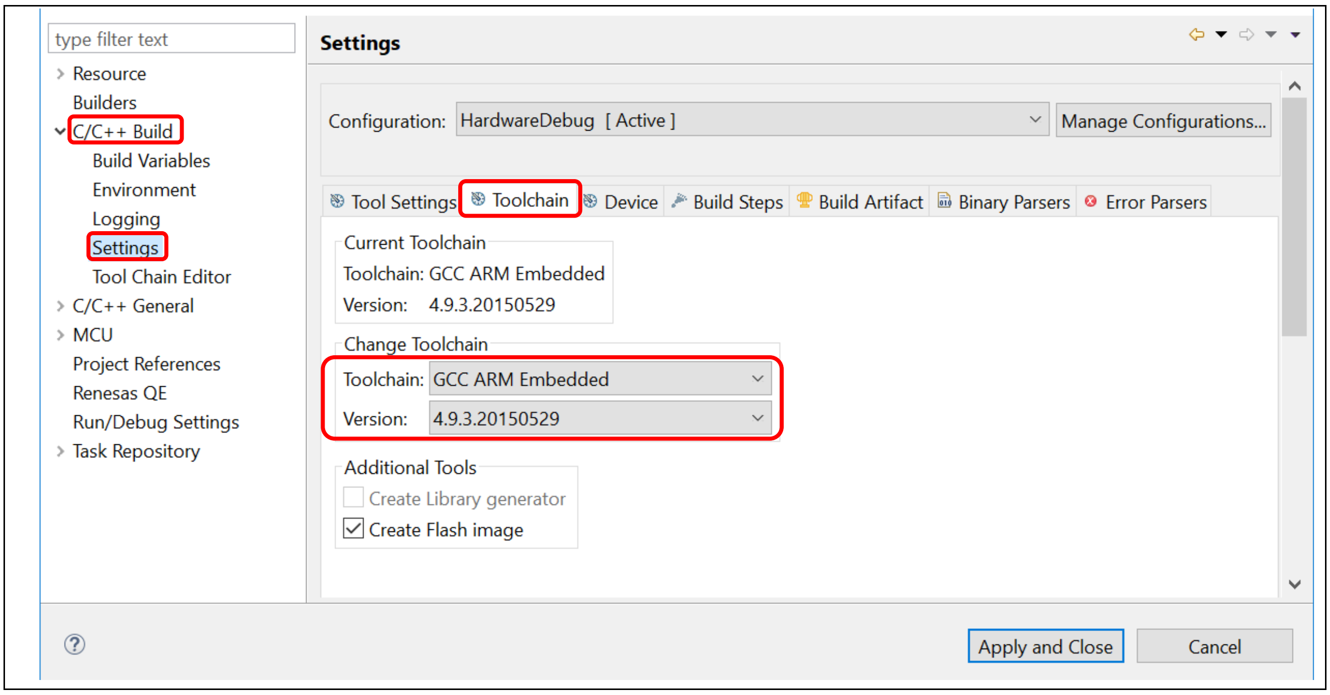


Figure 4-1. Opening the Properties Window

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



Configuration: HardwareDebug [Active]

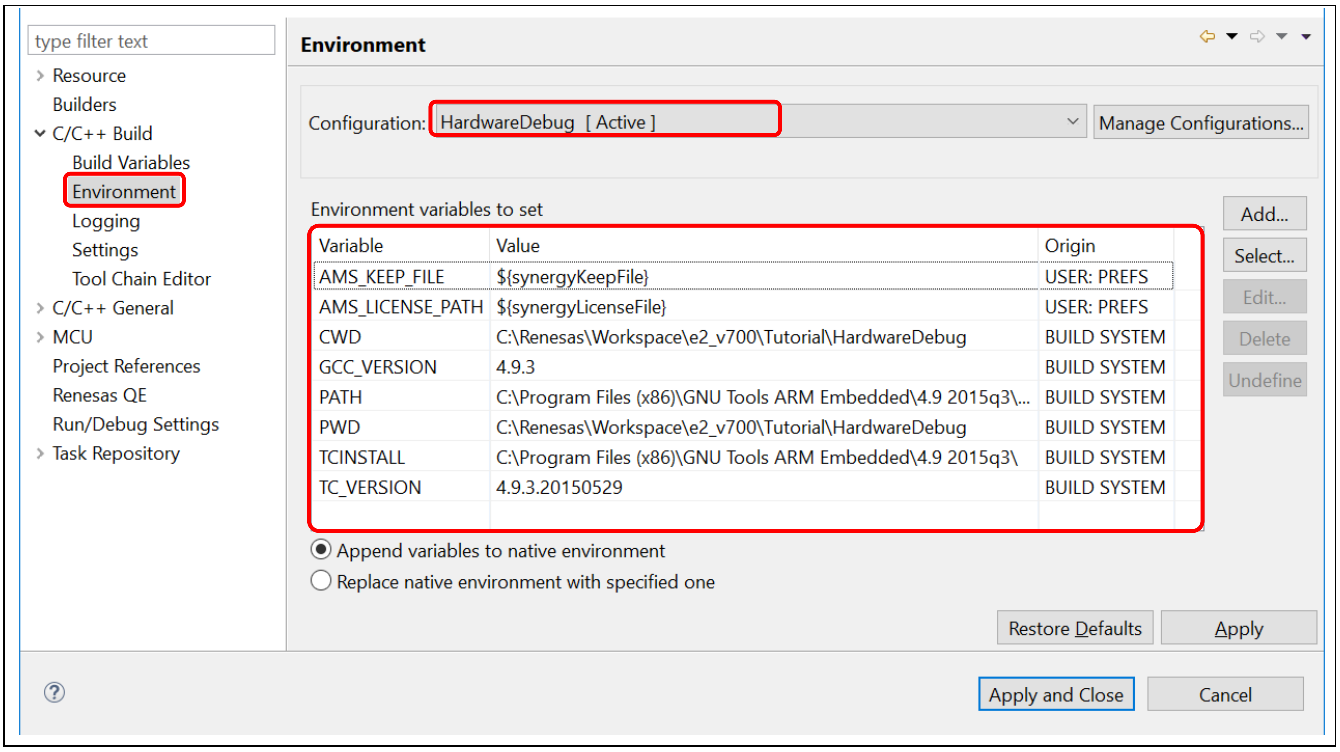
Current Toolchain
Toolchain: GCC ARM Embedded
Version: 4.9.3.20150529

Change Toolchain
Toolchain: GCC ARM Embedded
Version: 4.9.3.20150529

Additional Tools
☐ Create Library generator
☑ Create Flash image

Figure 4-2. Change Toolchain Version

3. Click **C/C++ Build → Environment** to select the current build configuration and add or edit environment variables.



Configuration: HardwareDebug [Active]

Environment variables to set

Variable	Value	Origin
AMS_KEEP_FILE	${synergyKeepFile}	USER: PREFS
AMS_LICENSE_PATH	${synergyLicenseFile}	USER: PREFS
CWD	C:\Renesas\Workspace\e2_v700\Tutorial\HardwareDebug	BUILD SYSTEM
GCC_VERSION	4.9.3	BUILD SYSTEM
PATH	C:\Program Files (x86)\GNU Tools ARM Embedded\4.9 2015q3\...	BUILD SYSTEM
PWD	C:\Renesas\Workspace\e2_v700\Tutorial\HardwareDebug	BUILD SYSTEM
TCINSTALL	C:\Program Files (x86)\GNU Tools ARM Embedded\4.9 2015q3\	BUILD SYSTEM
TC_VERSION	4.9.3.20150529	BUILD SYSTEM

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

Figure 4-3. Build Settings for Compiler: Environment

4. The build option allows user to retain all the toolchain configuration settings, including the path name specified by the environment variables. The current build configuration is **HardwareDebug**, as shown in Figure 4-3.

Build option details are described in the compiler user manual which is stored at `{Compiler installation directory}\share\doc`. For example, it can be found in `C:\Program Files (x86)\GNU Tools ARM Embedded\4.9 2015q3\share\doc`.

Note: As seen above, there is a **Tool Chain Editor** setting under **C/C++ Build**, but please do not change the configuration in this option. The **Tool Chain Editor** is used for toolchains which are NOT supported by Renesas build support plugins.

4.2 Building the Sample Project

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

1. Right click on the project and select **Build Project**
2. Click on the project to set focus and select **Project → Build Project**
3. Click on the project to set focus and click on the 🔨 icon
4. Click on the project to set focus and press **Ctrl** + **B**

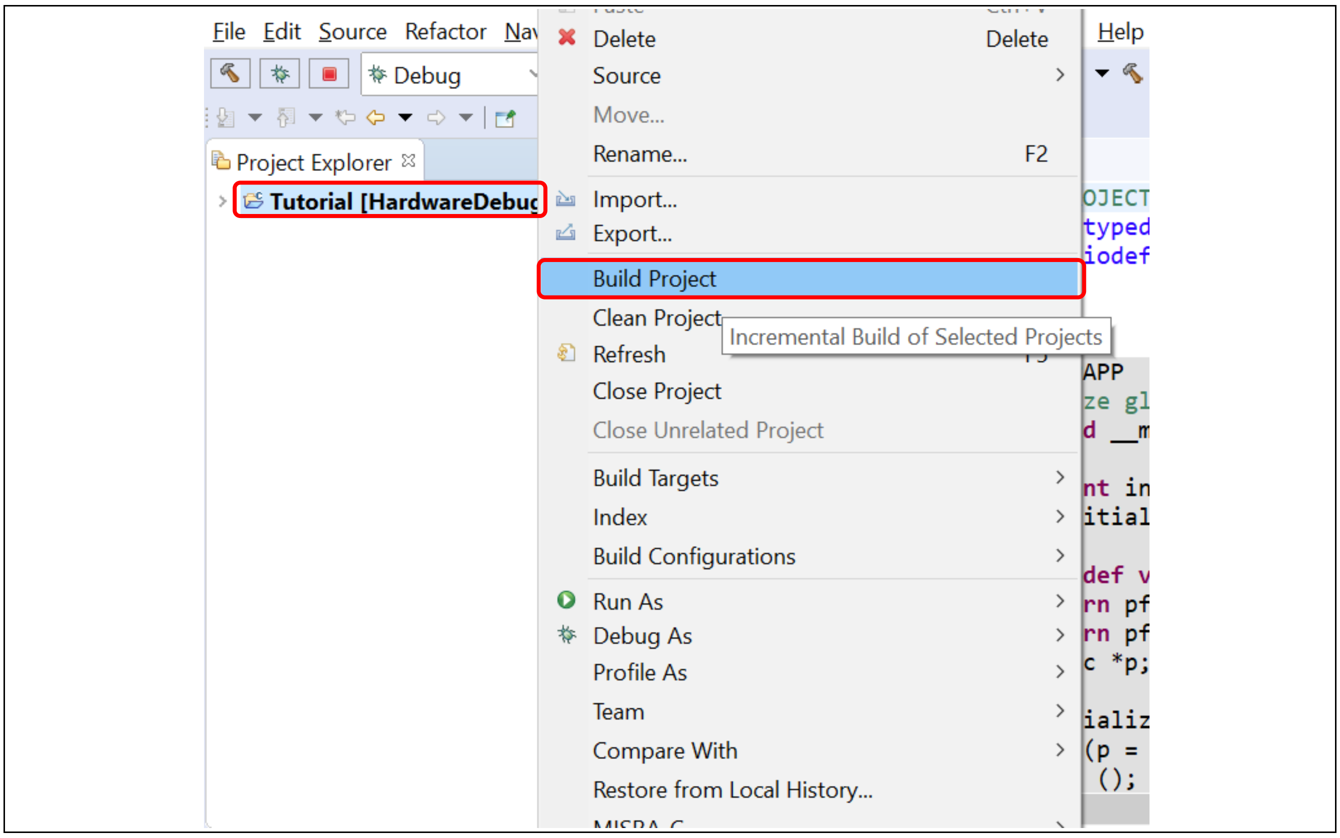
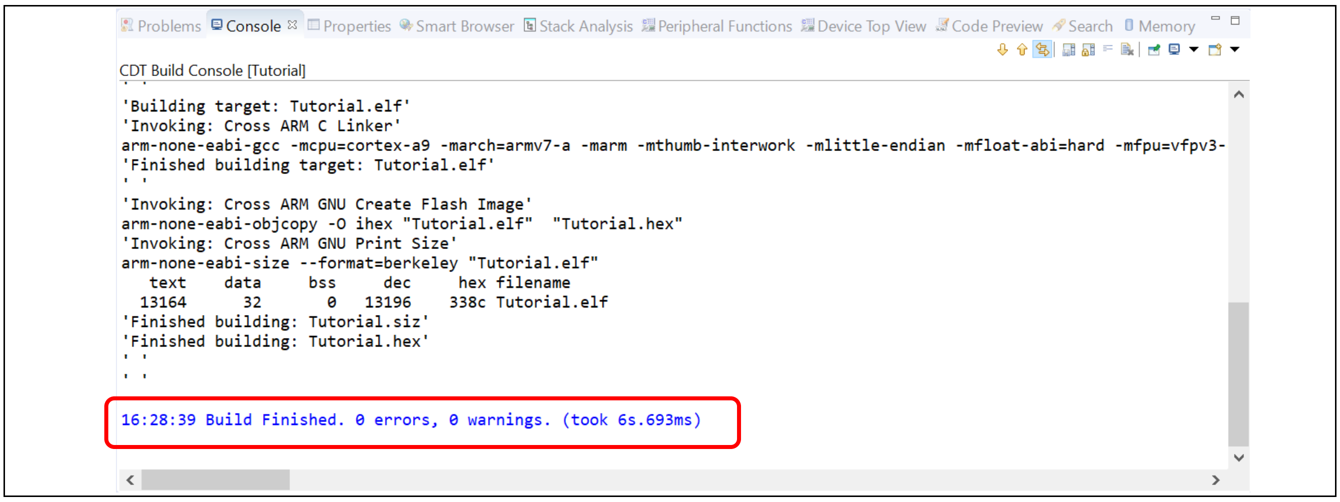


Figure 4-4. Building the Sample 'Tutorial' Project

5. The **Console** pane shows a **Build Finished** message to indicate a successful build. At the end of this build, files output to the ${CWD} directory consists of `makefile`, `Tutorial.elf`, `Tutorial.map`, `Tutorial.hex`, etc.

6. `Tutorial.elf` is the standard load module in ELF/DWARF format used for debugging.



```
'Building target: Tutorial.elf'
'Invoking: Cross ARM C Linker'
arm-none-eabi-gcc -mcpu=cortex-a9 -march=armv7-a -marm -mthumb-interwork -mlittle-endian -mfloat-abi=hard -mfpu=vfpv3-
'Finished building target: Tutorial.elf'
' '
'Invoking: Cross ARM GNU Create Flash Image'
arm-none-eabi-objcopy -O ihex "Tutorial.elf"  "Tutorial.hex"
'Invoking: Cross ARM GNU Print Size'
arm-none-eabi-size --format=berkeley "Tutorial.elf"
   text    data     bss     dec     hex filename
  13164      32       0   13196    338c Tutorial.elf
'Finished building: Tutorial.siz'
'Finished building: Tutorial.hex'
' '
' '

16:28:39 Build Finished. 0 errors, 0 warnings. (took 6s.693ms)
```

Figure 4-5. Project Has Built Successfully

4.3 Exporting Build Configuration Settings

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

1. Right-click in the **Project Explorer** pane to bring up the context menu
2. Select **Save build settings report** to save the build settings report

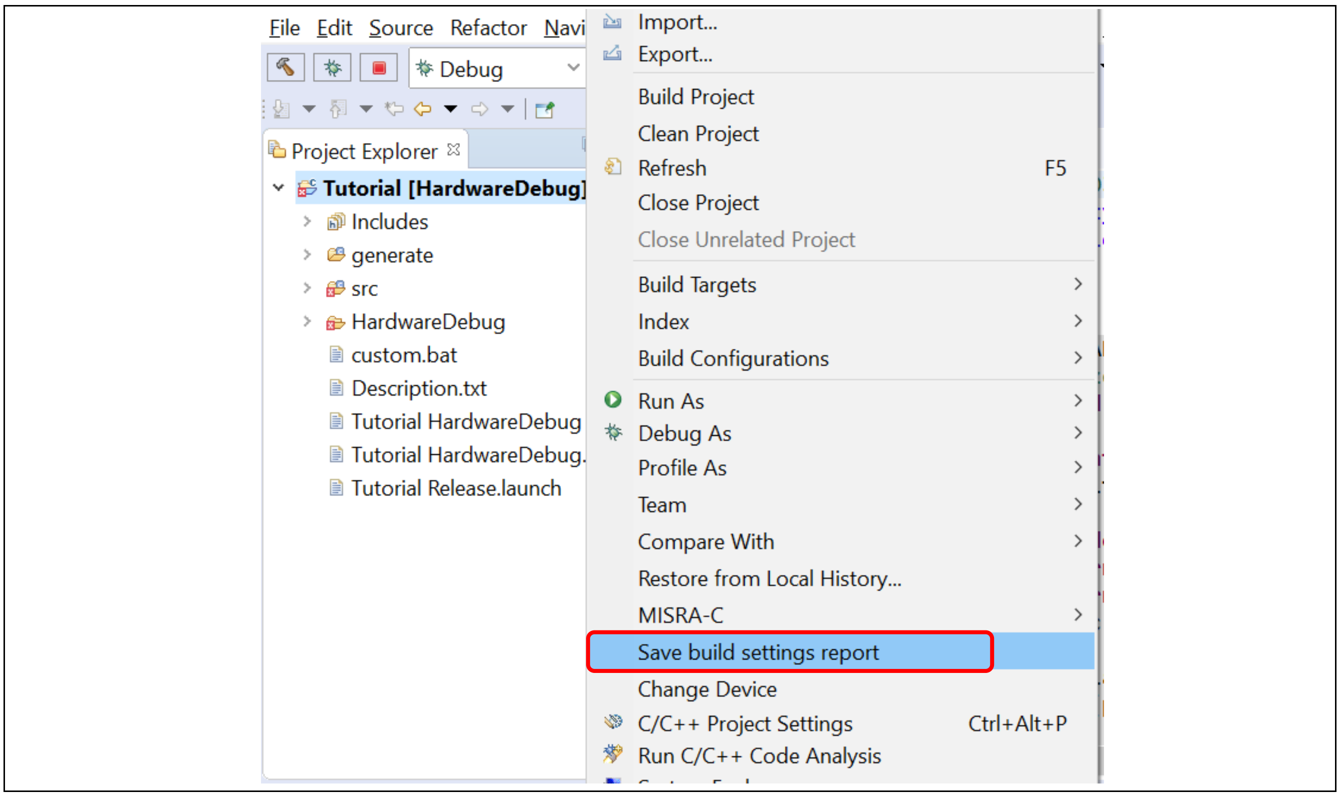


Figure 4-6. Project Reporter

5. Debug

This chapter describes the operation of the debug configuration and key debugging features of e^2 studio. The following illustration refers to 'Tutorial' project built (in Chapter 4.2) and is based on the hardware configuration of a Segger J-Link emulator and an RSK RZ/A1H board.

To open the specific perspective used for debugging, open 'Tutorial' project workspace in e^2 studio IDE and select the **Debug** perspective.

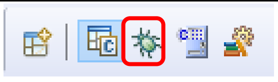


Figure 5-1. Switch to Debug Perspective

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

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

One or more perspectives can exist in a single Workbench window. Each perspective can be customized, and new ones can be added.

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

5.1 Change Existing Debug Configurations

The debug configuration has to be configured when debugging for the first time and only 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 the focus.
2. Click **Run** → **Debug Configurations…** or icon (downward arrow) → **Debug Configurations…** to open the **Debug Configurations** window.

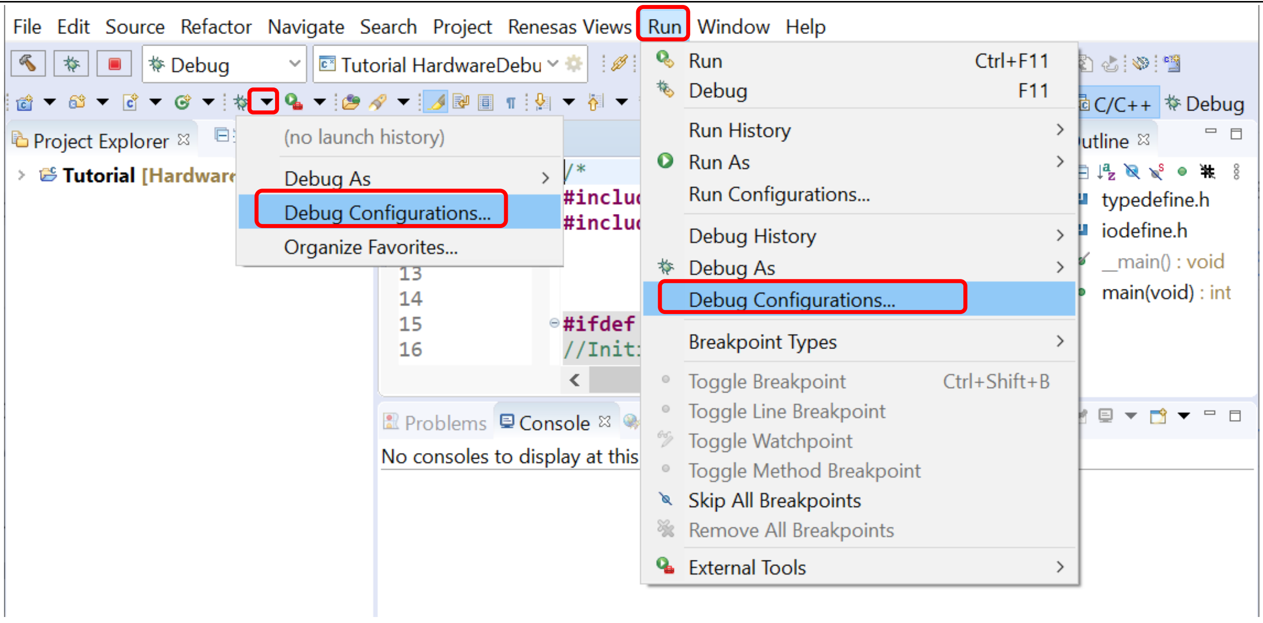
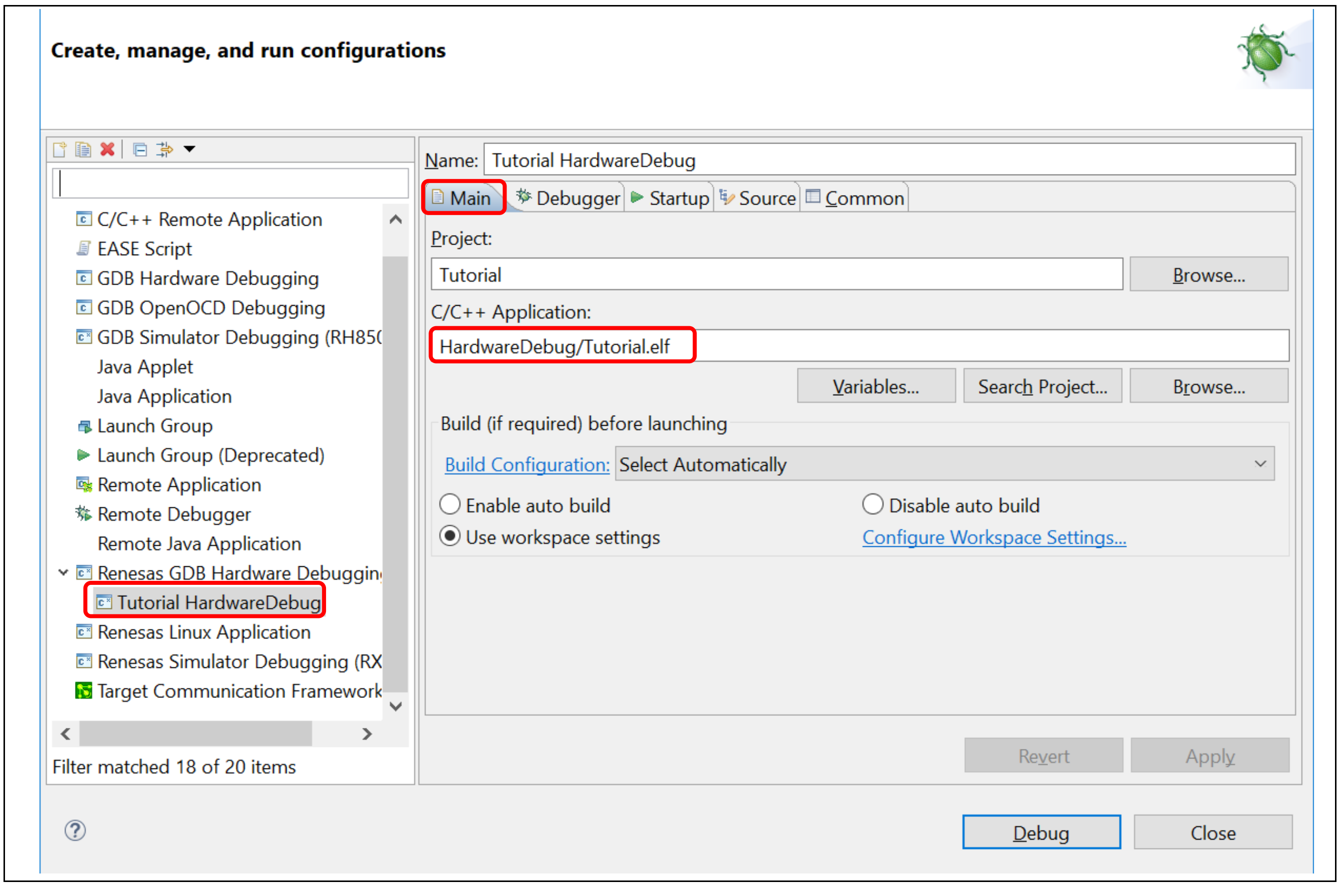


Figure 5-2. Open Debug Configurations Window

3. 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.elf`.


Create, manage, and run configurations

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

Filter matched 18 of 20 items

Name: Tutorial HardwareDebug

Main | Debugger | Startup | Source | Common

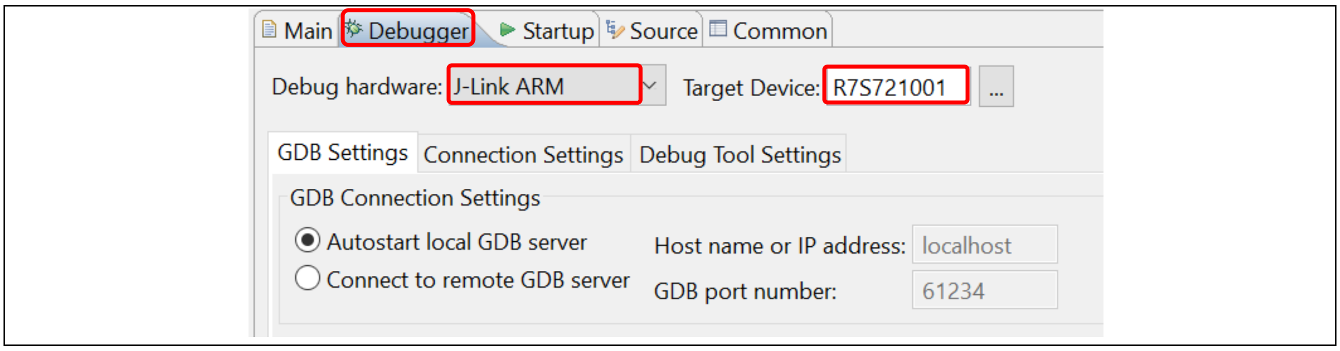
Project:
Tutorial

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

Build (if required) before launching
Build Configuration: Select Automatically
Enable auto build
Disable auto build
[✓ Use workspace settings]
Configure Workspace Settings...

Figure 5-3. Select Load Module

4. Switch to the **Debugger** tab, set 'J-Link ARM' as the debug hardware and 'R7S721001' as the target device.


Main | Debugger | Startup | Source | Common

Debug hardware: J-Link ARM
Target Device: R7S721001

GDB Settings | Connection Settings | Debug Tool Settings

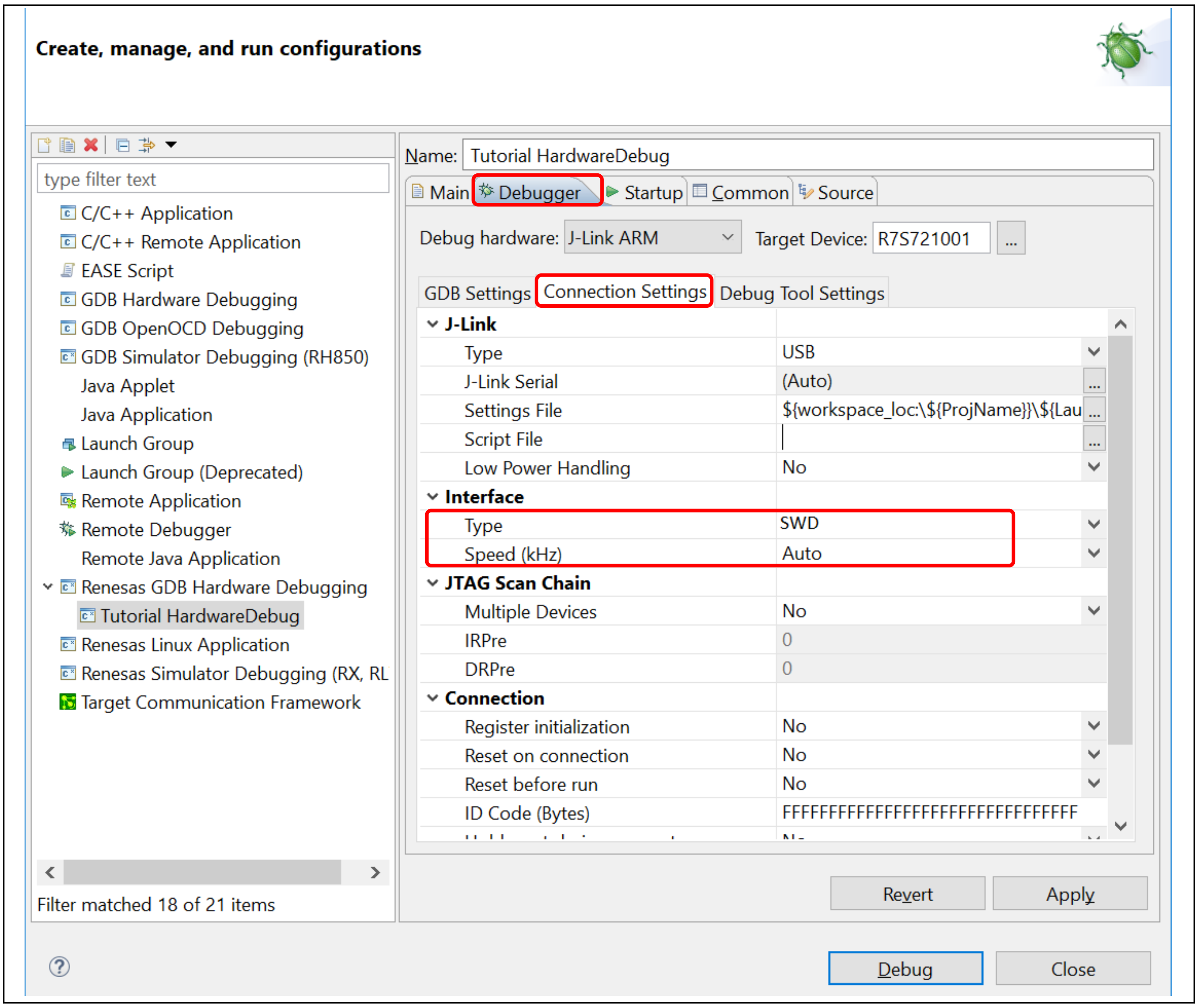
GDB Connection Settings
[✓ Autostart local GDB server]
Connect to remote GDB server
Host name or IP address: localhost
GDB port number: 61234

Figure 5-4. Select Target Device

5. Under the **Debugger** tab, go to the **Connection Settings** sub tab to configure the following based on the settings in Segger J-Link emulator and RSK RZ/A1H board:

   - Interface
     - Type = 'SWD'
     - Speed = 'Auto'

Note: The debug configuration shown in Figure 5-5 is an example. Incorrect settings may cause malfunction or damage to the hardware. So please verify the board and emulator settings carefully before connection.



Create, manage, and run configurations

type filter text

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

Filter matched 18 of 21 items

Name: Tutorial HardwareDebug

Main | Debugger | Startup | Common | Source

Debug hardware: J-Link ARM   Target Device: R7S721001

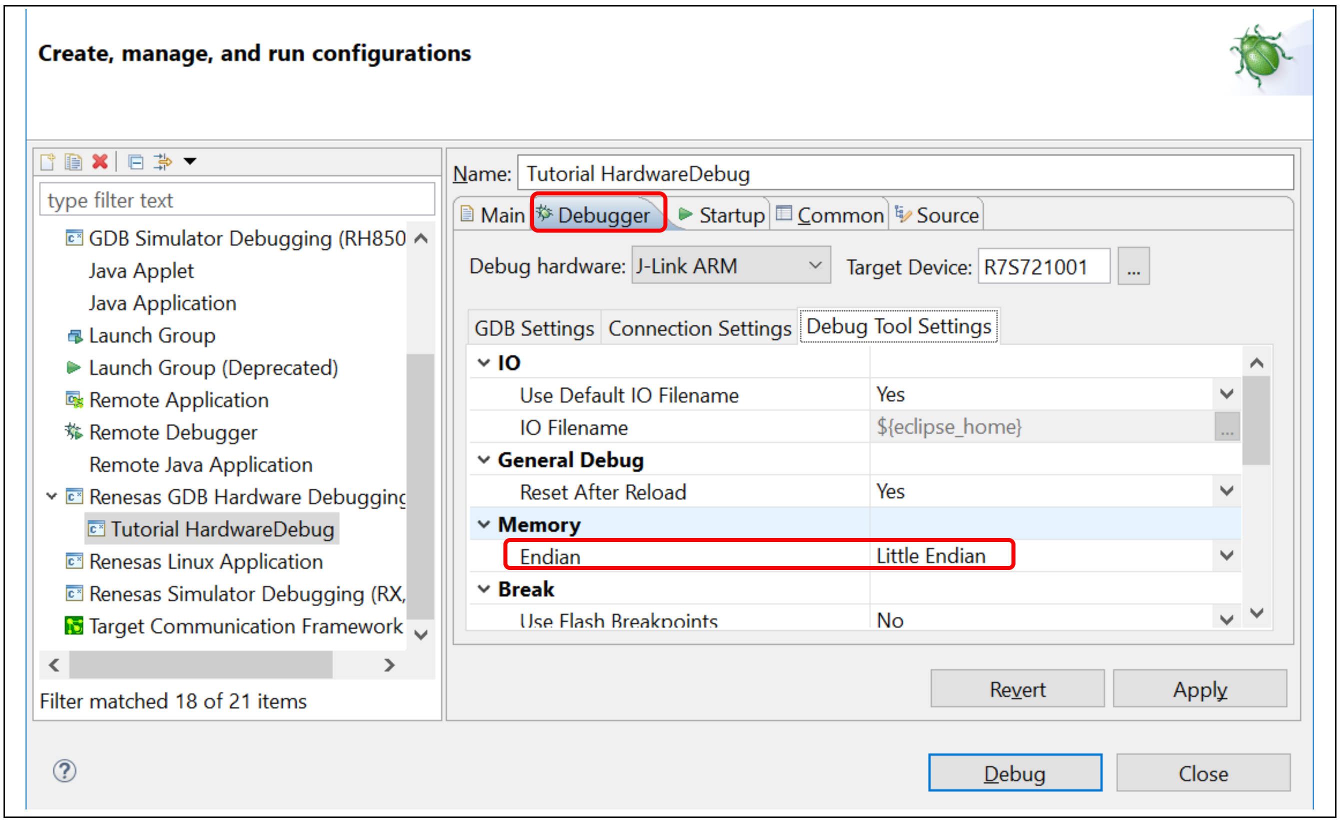
GDB Settings | Connection Settings | Debug Tool Settings

J-Link	
Type	USB
J-Link Serial	(Auto)
Settings File	${workspace_loc:\${ProjName}\${Lau
Script File	
Low Power Handling	No
Interface	
Type	SWD
Speed (kHz)	Auto
JTAG Scan Chain	
Multiple Devices	No
IRPre	0
DRPre	0
Connection	
Register initialization	No
Reset on connection	No
Reset before run	No
ID Code (Bytes)	FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

Figure 5-5. Change Connection Settings

6. Switch to the **Debug Tool Settings** sub tab, based on the RSK RZ/A1H board to ensure:

   - Memory
     Endian = 'Little Endian'



Create, manage, and run configurations

Name: Tutorial HardwareDebug

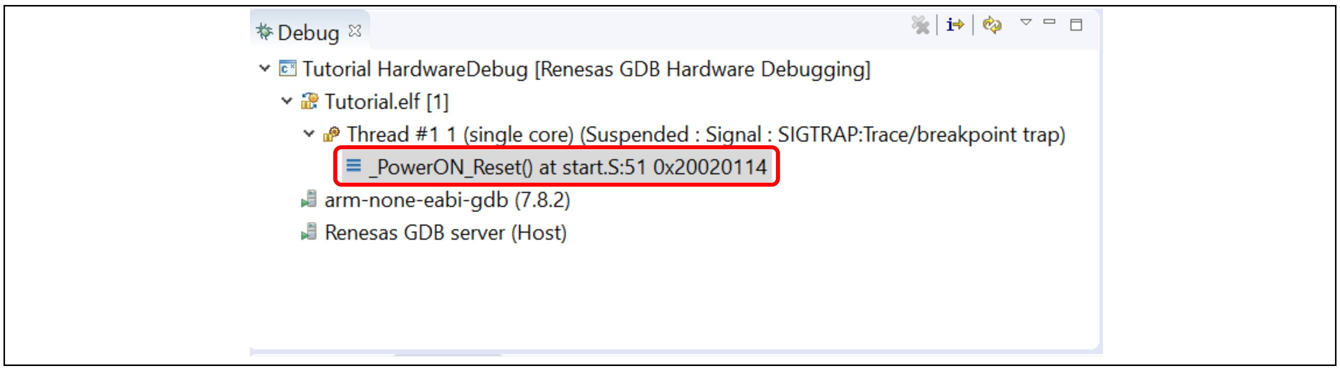
Debug hardware: J-Link ARM   Target Device: R7S721001

IO	
Use Default IO Filename	Yes
IO Filename	${eclipse_home}
General Debug	
Reset After Reload	Yes
Memory	
Endian	Little Endian
Break	
Use Flash Breakpoints	No

Figure 5-6. Change Debug Tool Settings

7. Click the **Apply** button to confirm the settings. Then click **Debug** to execute the debug launch configuration to connect to the Segger J-Link and RSK RZ/A1H board.

8. After the successful connection, select the **Debug** view to show target debugging information in a tree hierarchy. The program entry point is set to the function `_PowerON_Reset()` in file `start.S`.



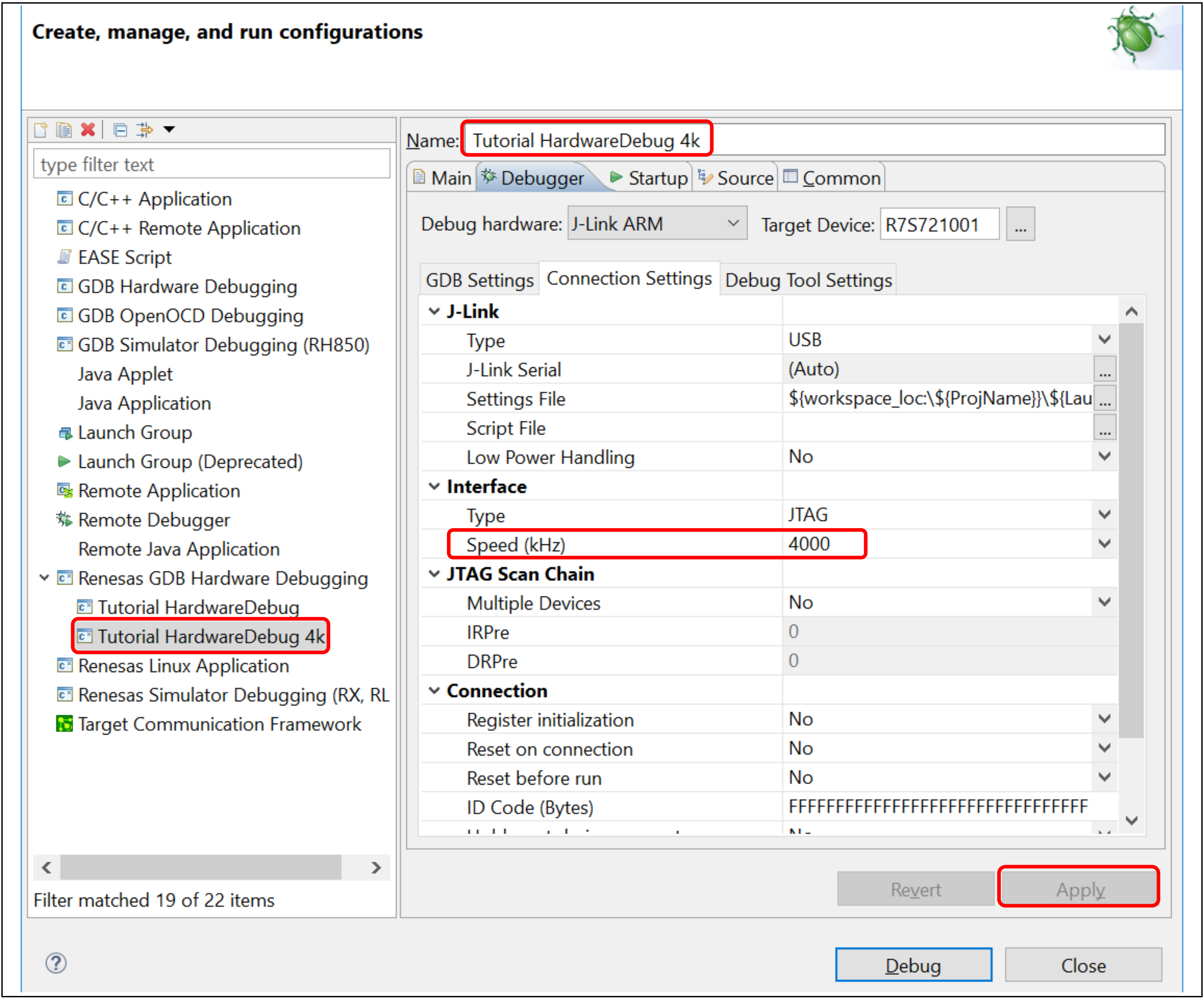
```
Debug
Tutorial HardwareDebug [Renesas GDB Hardware Debugging]
  Tutorial.elf [1]
    Thread #1 1 (single core) (Suspended : Signal : SIGTRAP:Trace/breakpoint trap)
      _PowerON_Reset() at start.S:51 0x20020114
  arm-none-eabi-gdb (7.8.2)
  Renesas GDB server (Host)
```

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 performed by the following steps.

1. Repeat step 1 in chapter 5.1 to open the **Debug Configurations** window.
2. Select a debug configuration (e.g. 'Tutorial HardwareDebug') and then click the icon (this duplicates the currently selected launch configuration). A new debug launch configuration (e.g. 'Tutorial HardwareDebug (1)') is created. It can be renamed to identify the settings by typing in the **Name** textbox and then clicking the **Apply** button.



Create, manage, and run configurations

type filter text

- C/C++ Application
- C/C++ Remote Application
- EASE Script
- GDB Hardware Debugging
- GDB OpenOCD Debugging
- GDB Simulator Debugging (RH850)
- Java Applet
- Java Application
- Launch Group
- Launch Group (Deprecated)
- Remote Application
- Remote Debugger
- Remote Java Application
- Renesas GDB Hardware Debugging
  - Tutorial HardwareDebug
  - Tutorial HardwareDebug 4k
- Renesas Linux Application
- Renesas Simulator Debugging (RX, RL
- Target Communication Framework

Filter matched 19 of 22 items

Name: Tutorial HardwareDebug 4k

Main | Debugger | Startup | Source | Common

Debug hardware: J-Link ARM   Target Device: R7S721001

GDB Settings | Connection Settings | Debug Tool Settings

J-Link	
Type	USB
J-Link Serial	(Auto)
Settings File	${workspace_loc:\${ProjName}\${Lau
Script File	
Low Power Handling	No
Interface	
Type	JTAG
Speed (kHz)	4000
JTAG Scan Chain	
Multiple Devices	No
IRPre	0
DRPre	0
Connection	
Register initialization	No
Reset on connection	No
Reset before run	No
ID Code (Bytes)	FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

Figure 5-8. Duplicate A Selected Debug Launch Configuration

3. The debug configuration can be configured as described in chapter 5.1. For example, change the interface speed to '4000'.

4. The launch configuration is marked with '[local]' and a '*' marker indicating that it is not yet attached to any project. Please specify project name in the **Common** tab.

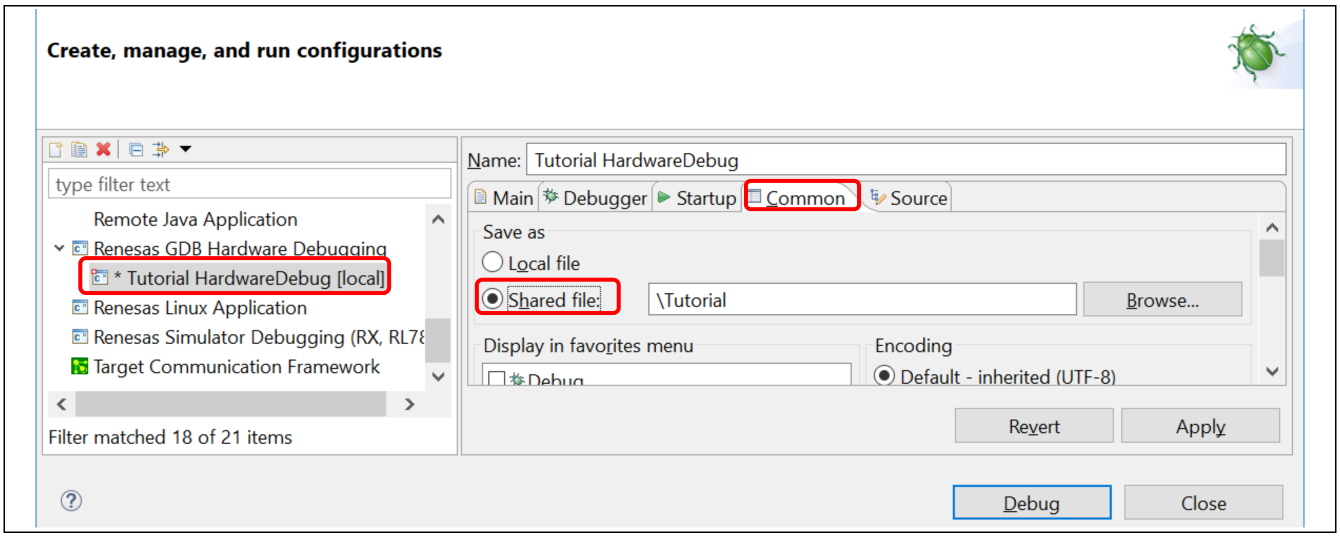

Create, manage, and run configurations
type filter text
Remote Java Application
Renesas GDB Hardware Debugging
* Tutorial HardwareDebug [local]
Renesas Linux Application
Renesas Simulator Debugging (RX, RL78
Target Communication Framework
Filter matched 18 of 21 items
Name: Tutorial HardwareDebug
Main
Debugger
Startup
Common
Source
Save as
Local file
Shared file:
\Tutorial
Browse...
Display in favorites menu
Debug
Encoding
Default - inherited (UTF-8)
Revert
Apply
Debug
Close

Figure 5-9. Attach Launch Configuration to a Specific Project

5.3 Launch Bar

This chapter explains the usage of the **Launch Bar**, which is supported from V6.0.0 or later.
The **Launch Bar** is located in the toolbar area of e^2 studio's main window.
This simple interface includes build and debug buttons for the selected launch target.

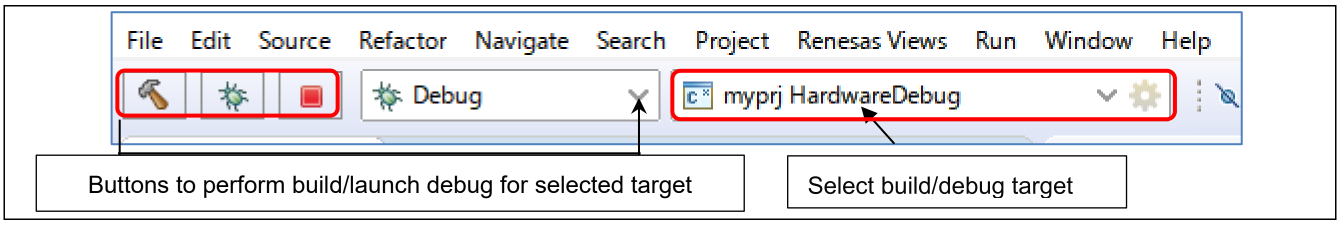


Figure 5-10. The Launch Bar Interface

The **Launch Bar** buttons behave as follows:

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

  Note: There is another build button in the project management toolbar which builds the active build configuration in **Project Explorer**. Note that the launch bar does not reflect the active build state of **Project Explorer**, so care must be taken to ensure that you're not building one configuration, and then launching another.

- The debug button triggers the launch of the debugger.
- The stop button terminates the selected launch target.

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

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

5.4 Basic Debugging Features

This chapter explains the debug views supported by the e^2 studio IDE.

- Standard GDB Debug (supported by Eclipse IDE framework): breakpoints, expressions, registers, memory, disassembly and variables
- Renesas Extension to Standard GDB Debug: eventpoints, I/O registers and trace.

The following are some useful buttons which exist in the **Debug** view:

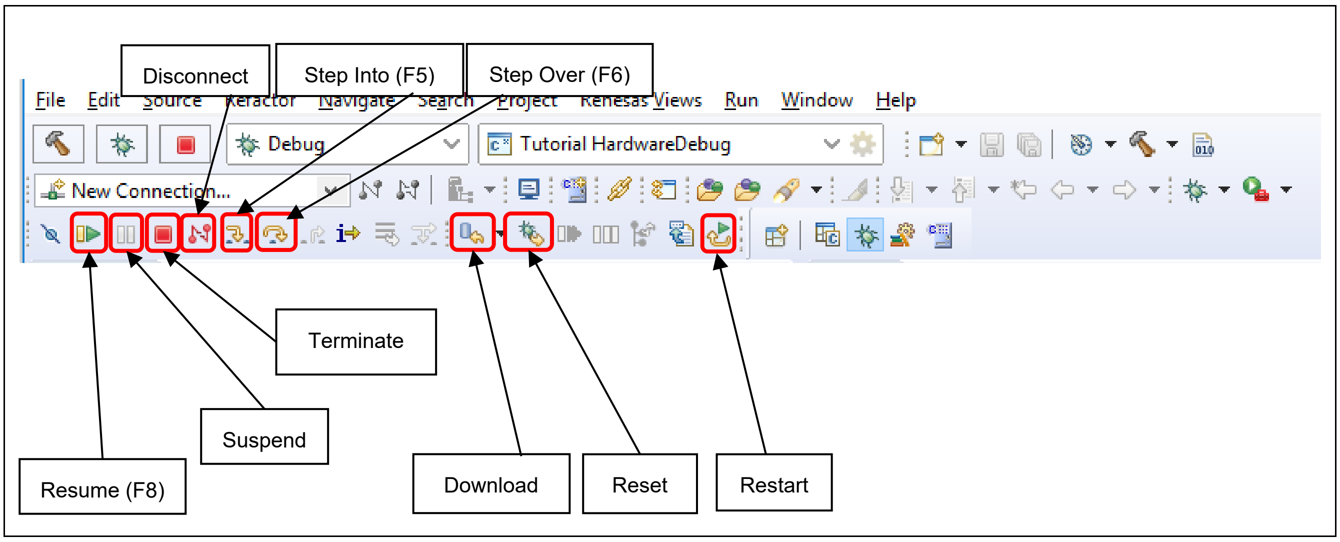


Figure 5-11. Useful Toolbars in Debug Views

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

The program can be paused on a breakpoint or by clicking the button. When the program is paused, you 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.

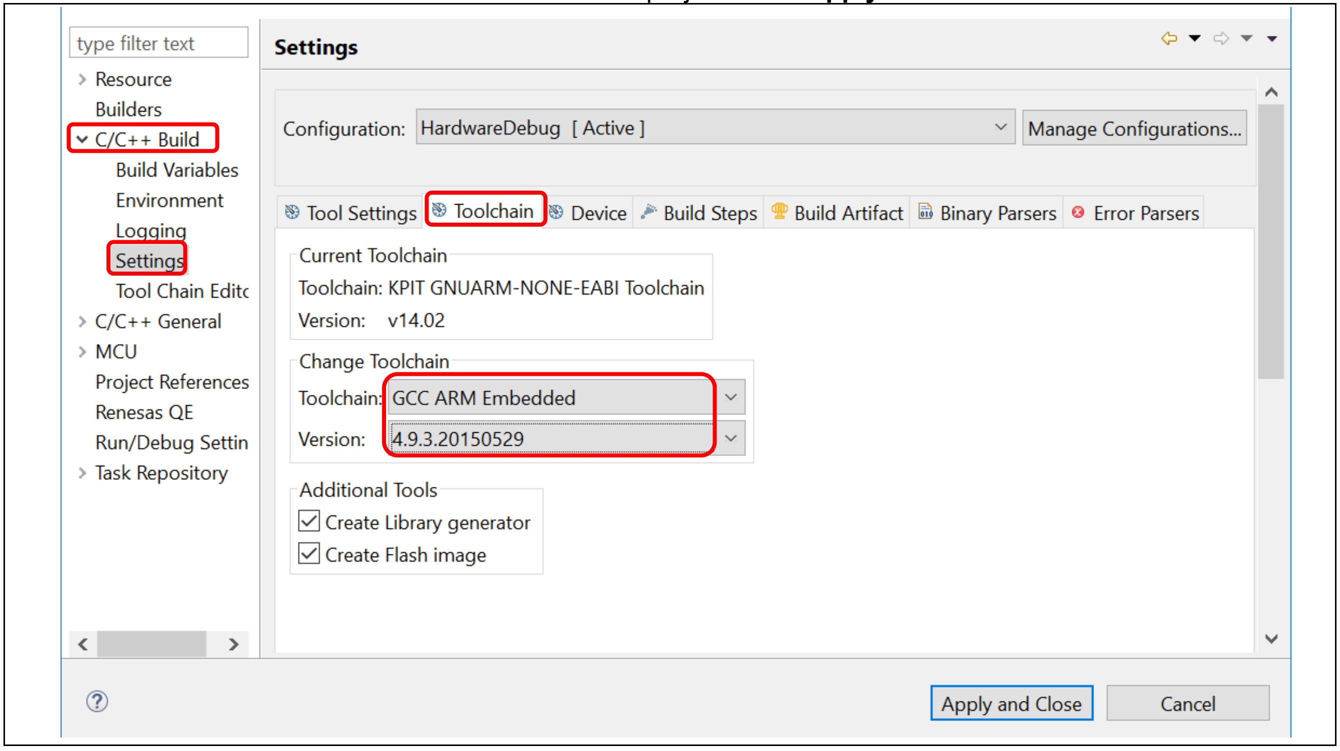
The button can be clicked to end the selected debug session and/or process, and the button can be clicked to disconnect the debugger from the selected process.

Some other operations are as following:

- The utton can be clicked to start new debug session.
- The button can be clicked to reset the program to the power-on-reset entry point.
- The button is used for re-downloading the binary file to the target system.

Note: To demonstate the features in following section, please use the sample code for the RZ/A1H from the Renesas website as follows:

1. Download the Starter Kit Sample Code for RZ/A1H from the Renesas website: https://www.renesas.com/sg/en/software/D6000665.html.
2. Follow the instructions in chapter 3.1 to import the project "RZ_A1H_Tutorial_RSK" to the workspace.
3. Open the project properties, select **C/C++ Build → Settings** in the left pane. Select tab **Toolchain** and select the GNU ARM Embedded toolchain for the project. Click **Apply and Close**.



Configuration: HardwareDebug [Active]

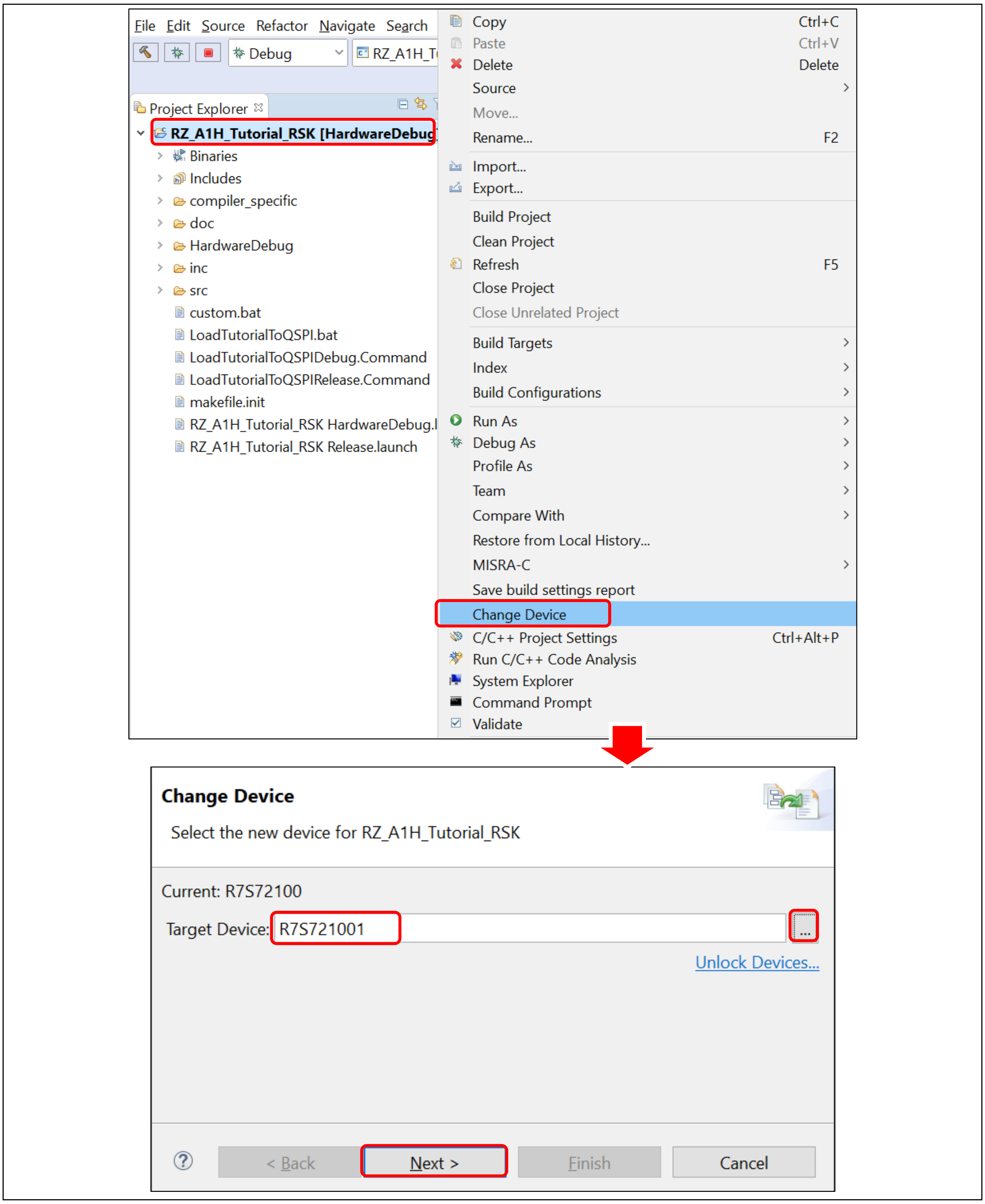
Current Toolchain
Toolchain: KPIT GNUARM-NONE-EABI Toolchain
Version: v14.02

Change Toolchain
Toolchain: GCC ARM Embedded
Version: 4.9.3.20150529

Additional Tools
☑ Create Library generator
☑ Create Flash image

Figure 5-12. Update Project Toolchain

4. Right click on the project name in **Project Explorer**, select **Change Device** to select the correct device.



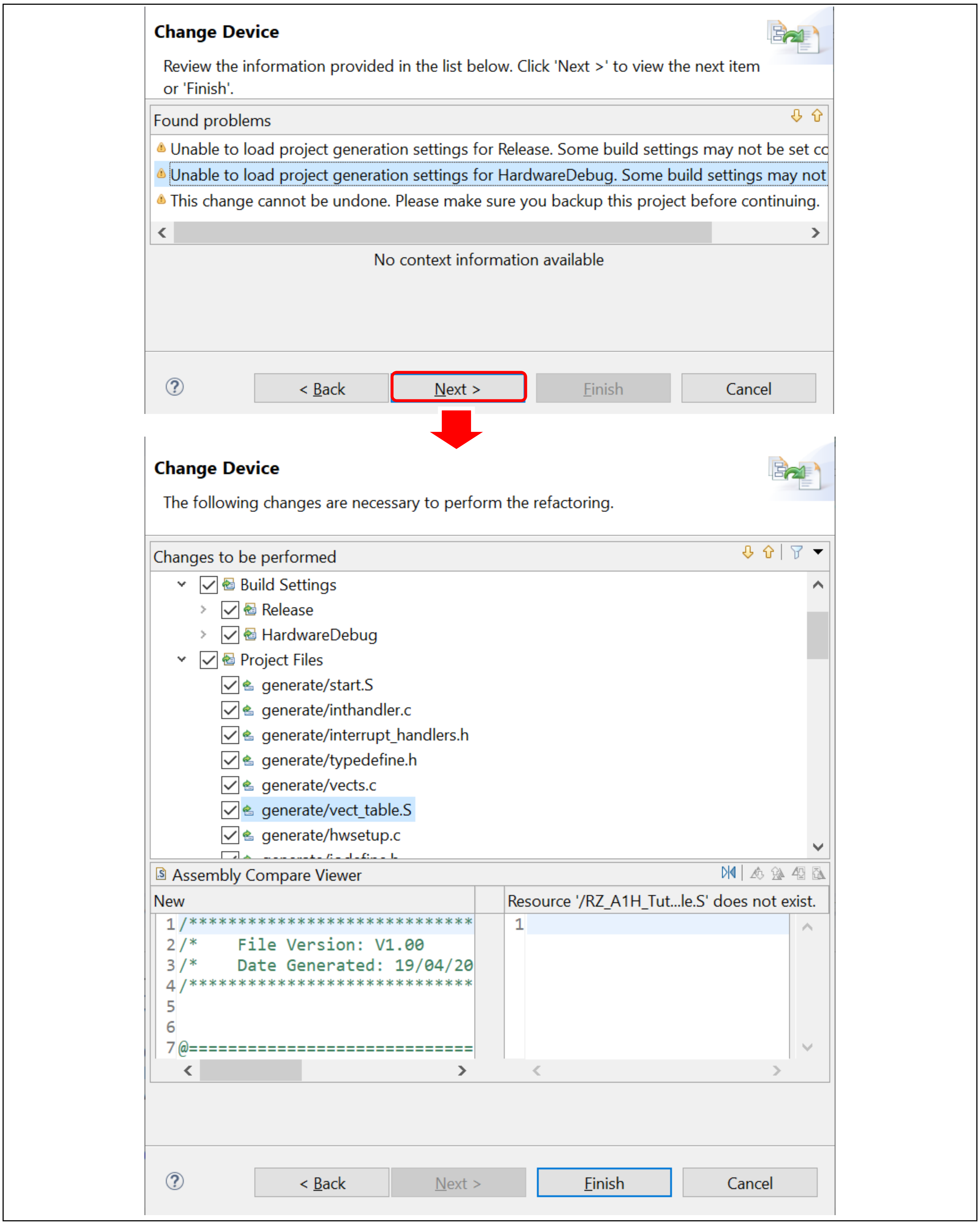
Select the new device for RZ_A1H_Tutorial_RSK

Current: R7S72100

Target Device: R7S721001

Figure 5-13. Change Device

5. After selecting the new device, click **Next** → **Next** to review changed information for the new device. If everything is OK, click **Finish**.



Change Device

Review the information provided in the list below. Click 'Next >' to view the next item or 'Finish'.

Found problems

Unable to load project generation settings for Release. Some build settings may not be set cc
Unable to load project generation settings for HardwareDebug. Some build settings may not
This change cannot be undone. Please make sure you backup this project before continuing.

No context information available

Change Device

The following changes are necessary to perform the refactoring.

Changes to be performed

- Build Settings
  - Release
  - HardwareDebug
- Project Files
  - generate/start.S
  - generate/inthandler.c
  - generate/interrupt_handlers.h
  - generate/typedefine.h
  - generate/vects.c
  - generate/vect_table.S
  - generate/hwsetup.c

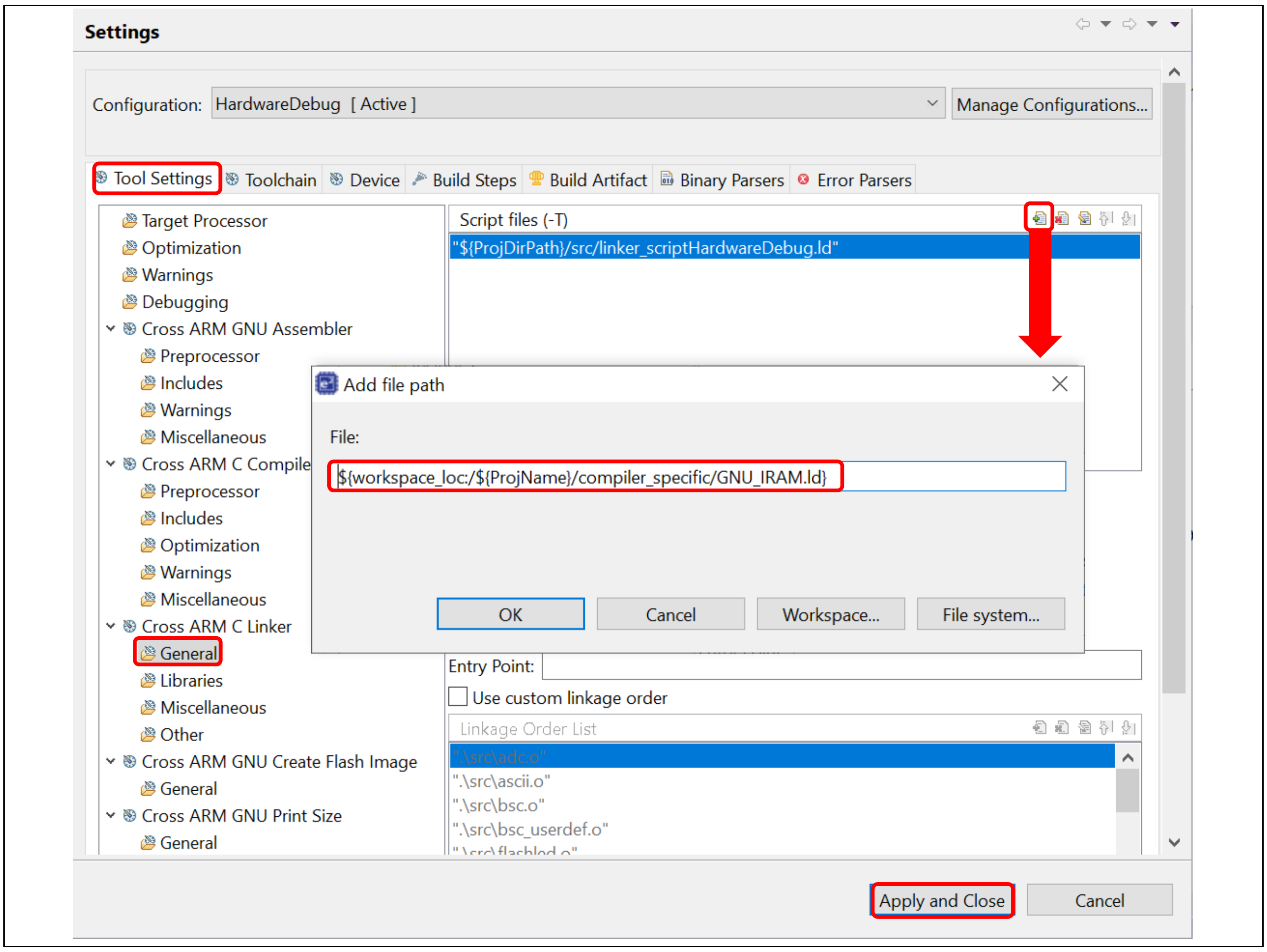
Assembly Compare Viewer

New | Resource '/RZ_A1H_Tut...le.S' does not exist.

```
/*****************************
/*     File Version: V1.00
/*     Date Generated: 19/04/20
/*****************************


@============================
```

Figure 5-14. Changed Information after Changing Device

6. Switch to the **Tool Settings** tab, select **Cross ARM C Linker → General**, and add the script file generated by the compiler to the script file list. Click **Apply and Close**.



Configuration: HardwareDebug [Active]

Script files (-T): "${ProjDirPath}/src/linker_scriptHardwareDebug.ld"

Add file path — File: ${workspace_loc:/${ProjName}/compiler_specific/GNU_IRAM.ld}

Linkage Order List:
".\src\adc.o"
".\src\ascii.o"
".\src\bsc.o"
".\src\bsc_userdef.o"

Figure 5-15. Add Compiler Generated Script File

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

5.4.1 Breakpoints View

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

To set the default breakpoint type (hardware or software):

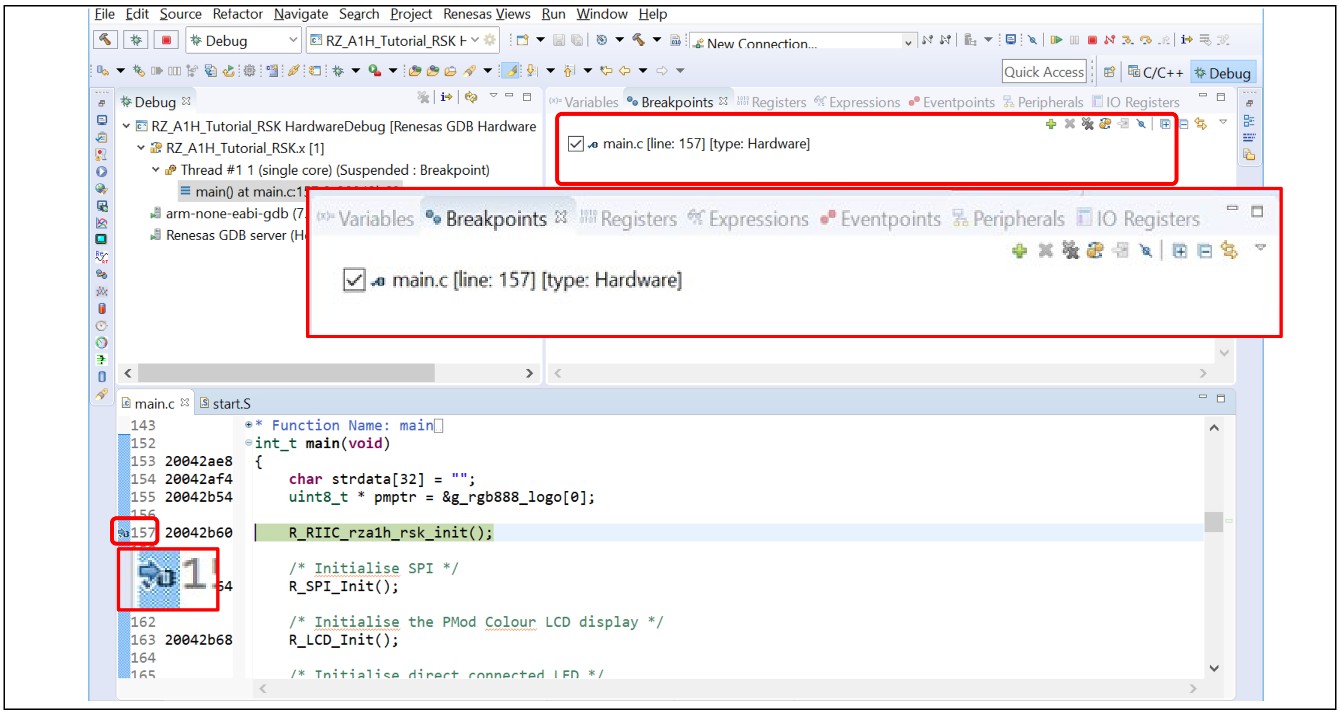
1. Right-click on the marker bar to bring up the context menu. For a hardware breakpoint, select **Breakpoint Types → e^2 studio Breakpoint**. For a software breakpoint, select **Breakpoint Types → C/C++ Breakpoints**.

To set a breakpoint:

1. Open `main.c` and double-click on the marker bar located in the left margin of the **C/C++ Editor** pane to set a breakpoint. A dot (hardware breakpoint) or (software breakpoint) is displayed in the marker bar depending on the **Breakpoint Type** selected. **Breakpoint Type** is hardware breakpoint by default.
2. Alternatively, right-click at the marker bar to choose **Toggle Hardware Breakpoint** or **Toggle Software Breakpoint** to set a hardware breakpoint or a software breakpoint .
3. Click **Windows → Show View → Breakpoints** or the breakpoints 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, you can choose to disable specific breakpoints or to skip all breakpoints:

1. To disable a specific breakpoint, right-click on the software breakpoint or hardware breakpoint located in the left margin of the **C/C++ Editor** pane and select **Disable Breakpoint**, or uncheck the related line in the breakpoints view. A disabled breakpoint is displayed as a white dot (or).
2. To skip all breakpoints, click on the icon in the breakpoints view. A blue dot with a backslash will appear in the editor pane as well as in the breakpoints view.



RZ_A1H_Tutorial_RSK HardwareDebug [Renesas GDB Hardware
RZ_A1H_Tutorial_RSK.x [1]
Thread #1 1 (single core) (Suspended : Breakpoint)
main() at main.c:1
arm-none-eabi-gdb (7.
Renesas GDB server (H

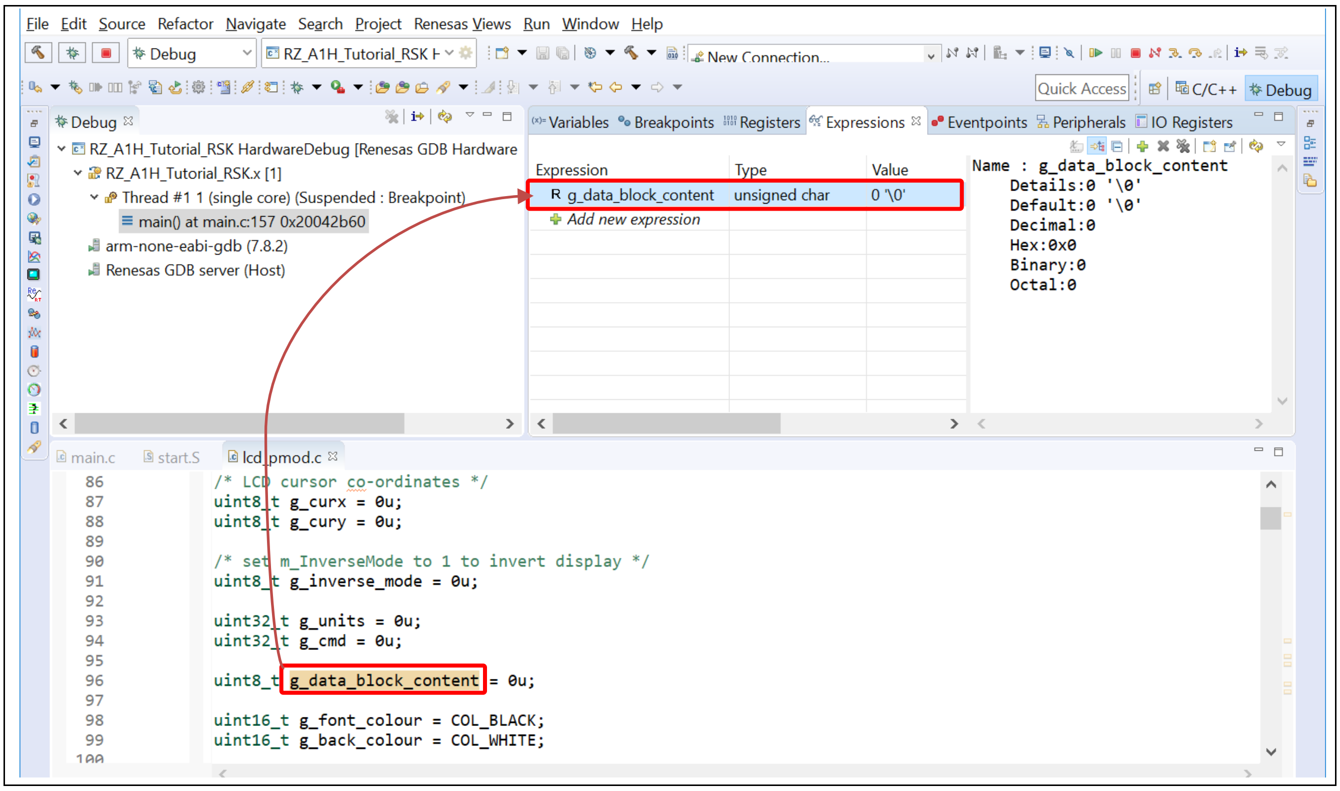
☑ main.c [line: 157] [type: Hardware]

```
143           * Function Name: main
152           int_t main(void)
153 20042ae8  {
154 20042af4      char strdata[32] = "";
155 20042b54      uint8_t * pmptr = &g_rgb888_logo[0];
156
157 20042b60      R_RIIC_rza1h_rsk_init();

                  /* Initialise SPI */
            54    R_SPI_Init();

162               /* Initialise the PMod Colour LCD display */
163 20042b68      R_LCD_Init();
164
165               /* Initialise direct connected LED */
```

Figure 5-16. Breakpoints view

5.4.2 Expressions View

The **Expressions** view monitors the value of global variables, static variables or local variables during debugging. For all RZ/A1H debuggers, these variables (including the local variables in scope) can be set for real-time refresh.


File Edit Source Refactor Navigate Search Project Renesas Views Run Window Help
Debug
RZ_A1H_Tutorial_RSK H
New Connection...
Quick Access
C/C++
Debug
Debug
RZ_A1H_Tutorial_RSK HardwareDebug [Renesas GDB Hardware
RZ_A1H_Tutorial_RSK.x [1]
Thread #1 1 (single core) (Suspended : Breakpoint)
main() at main.c:157 0x20042b60
arm-none-eabi-gdb (7.8.2)
Renesas GDB server (Host)
Variables
Breakpoints
Registers
Expressions
Eventpoints
Peripherals
IO Registers
Add new expression
main.c
start.S
lcd_pmod.c

Expression	Type	Value
R g_data_block_content	unsigned char	0 '\0'

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

```
/* LCD cursor co-ordinates */
uint8_t g_curx = 0u;
uint8_t g_cury = 0u;

/* set m_InverseMode to 1 to invert display */
uint8_t g_inverse_mode = 0u;

uint32_t g_units = 0u;
uint32_t g_cmd = 0u;

uint8_t g_data_block_content = 0u;

uint16_t g_font_colour = COL_BLACK;
uint16_t g_back_colour = COL_WHITE;
```

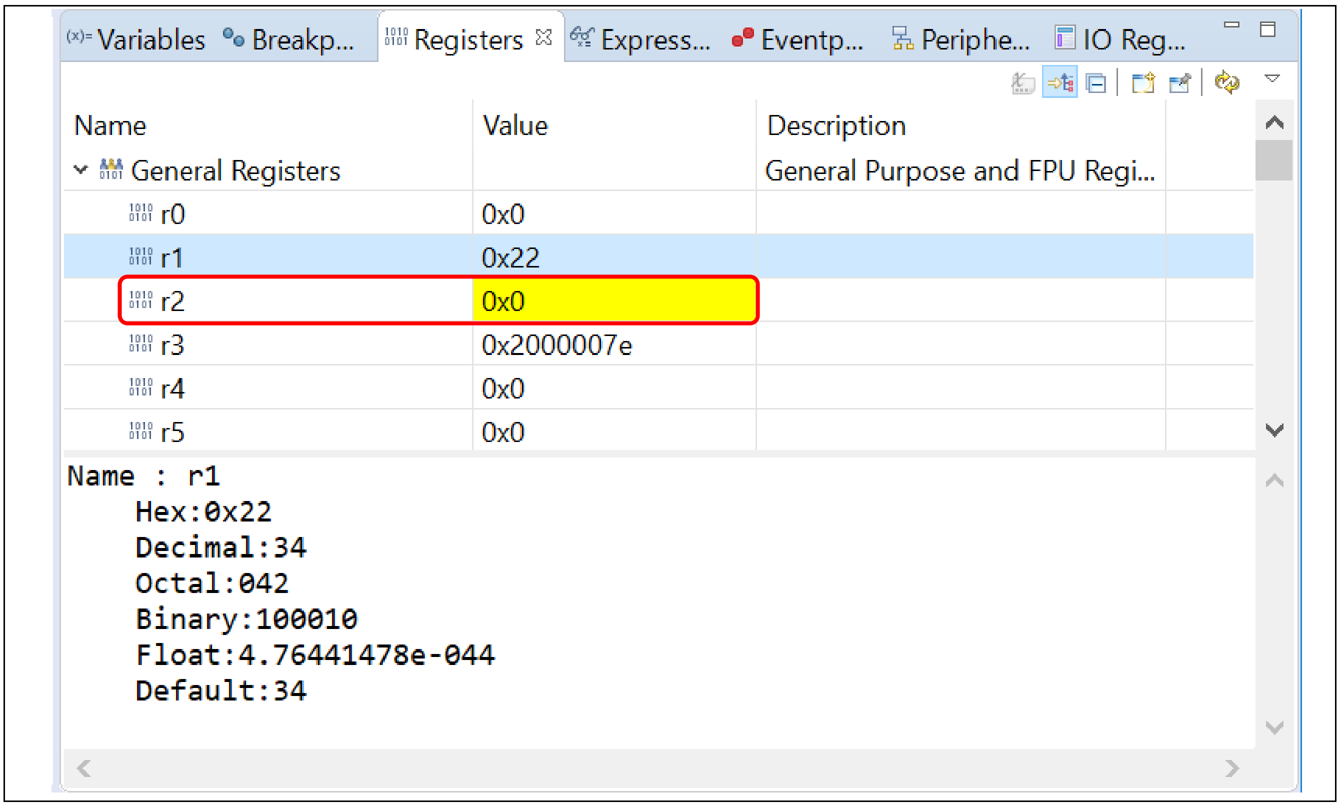
Figure 5-17. Expressions View

To watch a global variable:

1. Click **Window → Show View → Expressions** or the icon to open the **Expressions** view
2. Drag and drop a global variable over to the **Expressions** view. Alternatively, right-click on the global variable and 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 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.

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.



Name	Value	Description
General Registers		General Purpose and FPU Regi...
r0	0x0	
r1	0x22	
r2	0x0	
r3	0x2000007e	
r4	0x0	
r5	0x0	

```
Name : r1
    Hex:0x22
    Decimal:34
    Octal:042
    Binary:100010
    Float:4.76441478e-044
    Default:34
```

Figure 5-18. Registers View

To view the general register 'r0':

1. Click **Window** → **Show View** → **Registers** or the icon to open the **Registers** view.
2. Click 'r0' to view the values in a different radix.

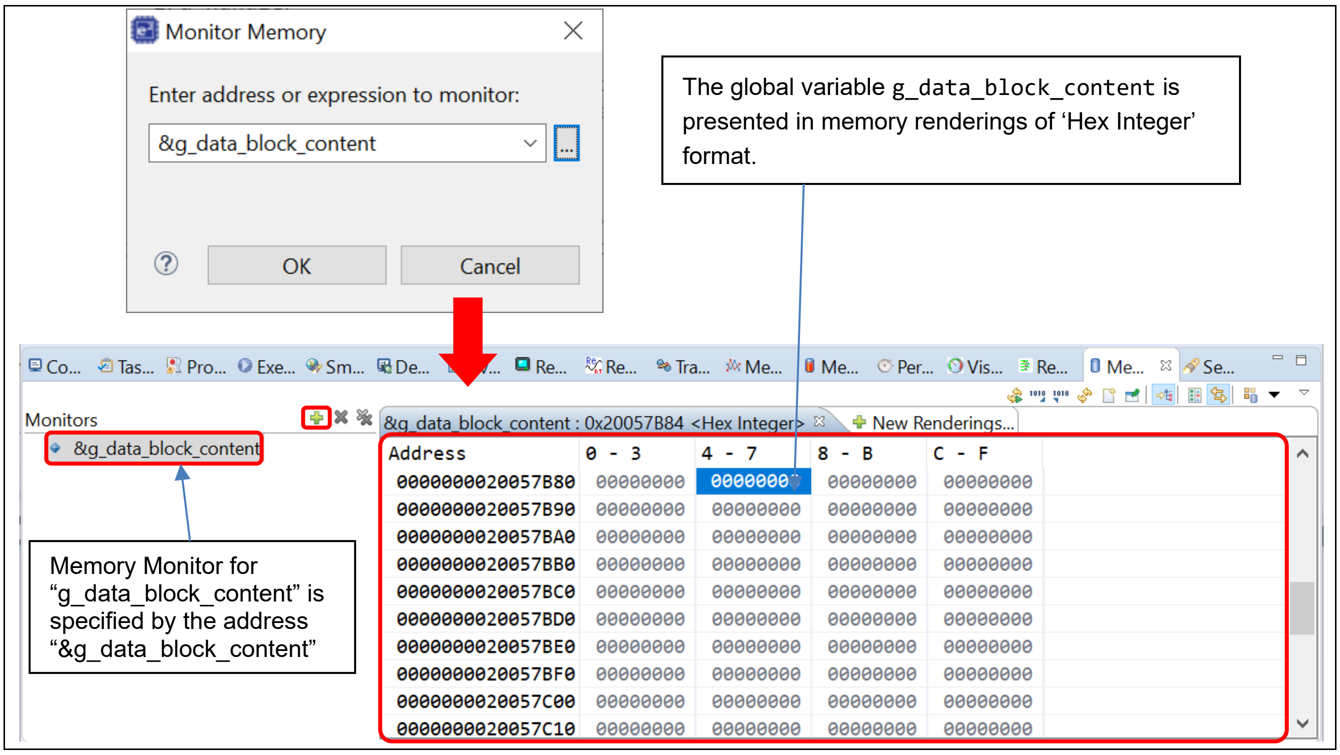
Values that have been changed are highlighted in yellow in the **Registers** view when the program stops.

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 base address. The memory data in each memory monitor can be presented in different 'memory renderings', which are predefined data formats (e.g. hex integer, signed integer, unsigned integer, ASCII, image, etc.)

To view the memory of a variable (e.g. `g_data_block_content`):

1. Click **Window** → **Show View** → **Memory** or the icon to open the **Memory** view.
2. Click the icon to open **Monitor Memory** dialog box. Enter the address of the variable `g_data_block_content`.


Monitor Memory
Enter address or expression to monitor:
&g_data_block_content
OK
Cancel
Co... Tas... Pro... Exe... Sm... De... ... Re... Re... Tra... Me... Me... Per... Vis... Re... Me... Se...
Monitors
&g_data_block_content
&g_data_block_content : 0x20057B84 <Hex Integer>
New Renderings...

The global variable `g_data_block_content` is presented in memory renderings of 'Hex Integer' format.

Memory Monitor for "g_data_block_content" is specified by the address "&g_data_block_content"

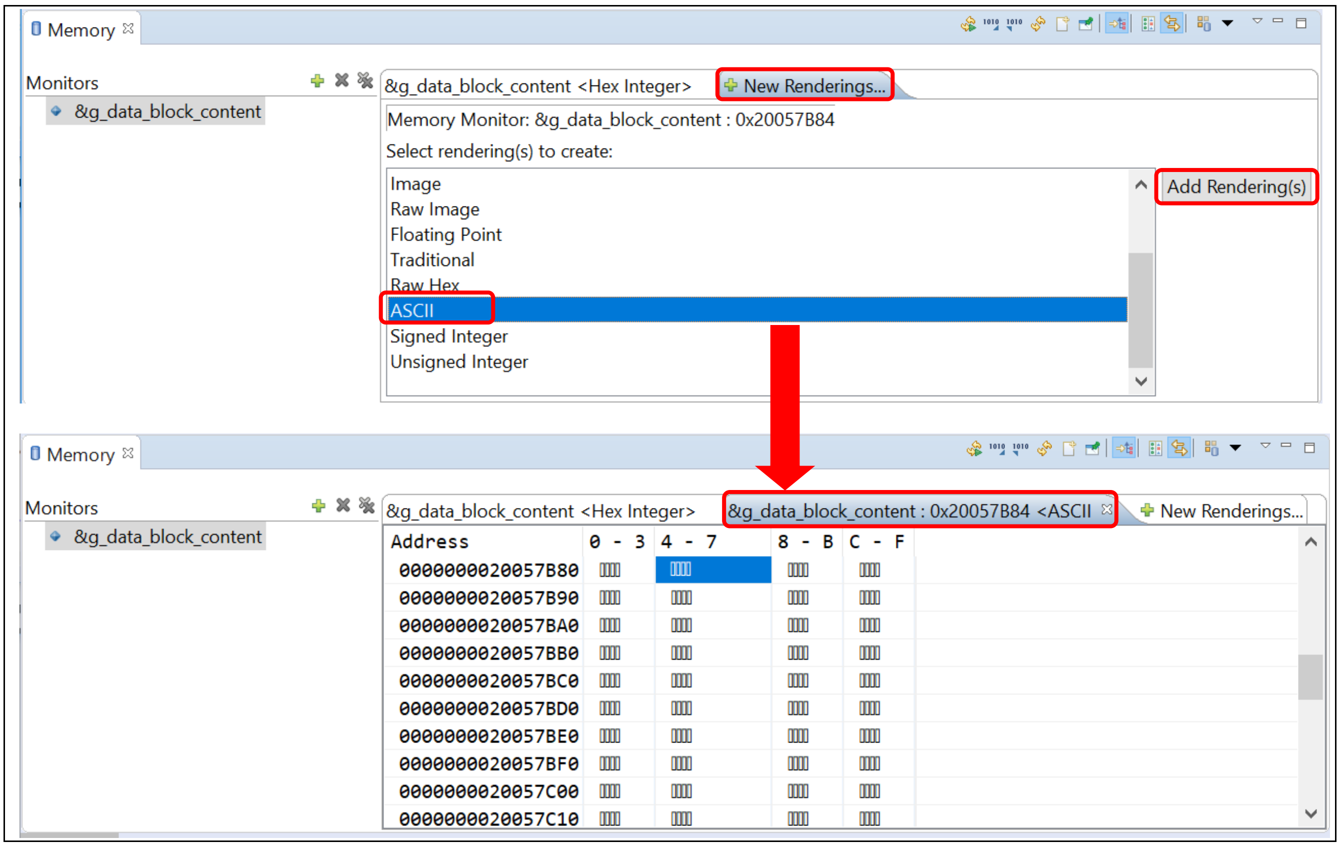
```
Address            0 - 3      4 - 7      8 - B      C - F
0000000020057B80   00000000   00000000   00000000   00000000
0000000020057B90   00000000   00000000   00000000   00000000
0000000020057BA0   00000000   00000000   00000000   00000000
0000000020057BB0   00000000   00000000   00000000   00000000
0000000020057BC0   00000000   00000000   00000000   00000000
0000000020057BD0   00000000   00000000   00000000   00000000
0000000020057BE0   00000000   00000000   00000000   00000000
0000000020057BF0   00000000   00000000   00000000   00000000
0000000020057C00   00000000   00000000   00000000   00000000
0000000020057C10   00000000   00000000   00000000   00000000
```

Figure 5-19. Memory View (1/2)

To add new renderings format (e.g. ASCII) for the variable g_data_block_content:

1. Click the New Renderings... tab and select 'ASCII' to add the rendering

   This creates a new tab named '&g_data_block_content < ASCII>' next to the tab '&g_data_block_content <Hex Integer>'.



Memory Monitor: &g_data_block_content : 0x20057B84

Select rendering(s) to create:

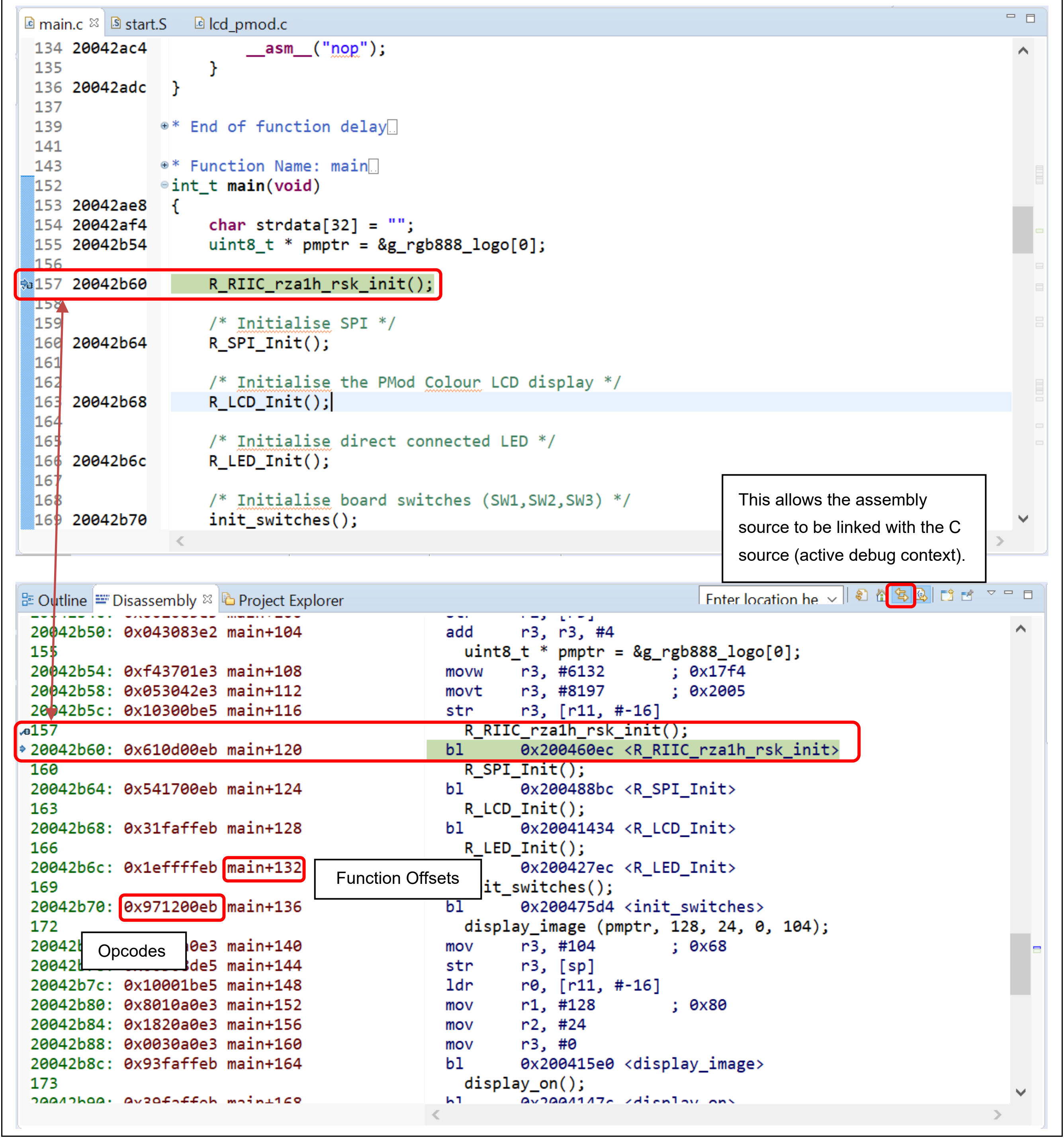
- Image
- Raw Image
- Floating Point
- Traditional
- Raw Hex
- ASCII
- Signed Integer
- Unsigned Integer

```
Address           0 - 3  4 - 7  8 - B  C - F
0000000020057B80  □□□□   □□□□   □□□□   □□□□
0000000020057B90  □□□□   □□□□   □□□□   □□□□
0000000020057BA0  □□□□   □□□□   □□□□   □□□□
0000000020057BB0  □□□□   □□□□   □□□□   □□□□
0000000020057BC0  □□□□   □□□□   □□□□   □□□□
0000000020057BD0  □□□□   □□□□   □□□□   □□□□
0000000020057BE0  □□□□   □□□□   □□□□   □□□□
0000000020057BF0  □□□□   □□□□   □□□□   □□□□
0000000020057C00  □□□□   □□□□   □□□□   □□□□
0000000020057C10  □□□□   □□□□   □□□□   □□□□
```

Figure 5-20. 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 currently executing line is highlighted by an arrow marker in the view. In the **Disassembly** view, user can set breakpoints on assembler instructions, enable or disable these breakpoints, step through the assembler instructions and even jump to a specific instruction in the program.



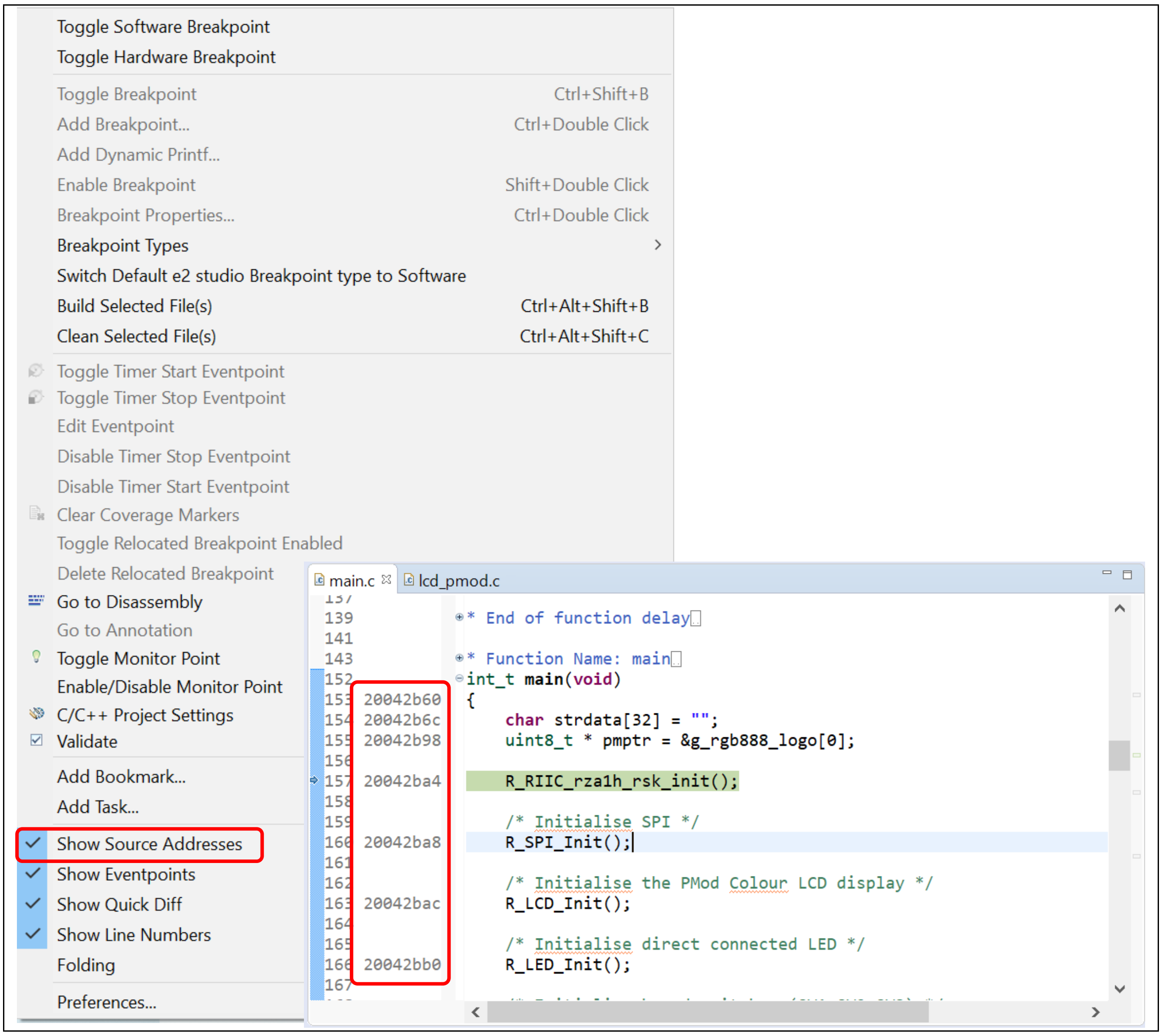
```
134 20042ac4          __asm__("nop");
135                 }
136 20042adc    }
137
139           * End of function delay
141
143           * Function Name: main
152          int_t main(void)
153 20042ae8    {
154 20042af4        char strdata[32] = "";
155 20042b54        uint8_t * pmptr = &g_rgb888_logo[0];
156
157 20042b60        R_RIIC_rza1h_rsk_init();
158
159                 /* Initialise SPI */
160 20042b64        R_SPI_Init();
161
162                 /* Initialise the PMod Colour LCD display */
163 20042b68        R_LCD_Init();
164
165                 /* Initialise direct connected LED */
166 20042b6c        R_LED_Init();
167
168                 /* Initialise board switches (SW1,SW2,SW3) */
169 20042b70        init_switches();
```

```
20042b50: 0x043083e2 main+104        add      r3, r3, #4
155                                    uint8_t * pmptr = &g_rgb888_logo[0];
20042b54: 0xf43701e3 main+108        movw     r3, #6132        ; 0x17f4
20042b58: 0x053042e3 main+112        movt     r3, #8197        ; 0x2005
20042b5c: 0x10300be5 main+116        str      r3, [r11, #-16]
157                                    R_RIIC_rza1h_rsk_init();
20042b60: 0x610d00eb main+120        bl       0x200460ec <R_RIIC_rza1h_rsk_init>
160                                    R_SPI_Init();
20042b64: 0x541700eb main+124        bl       0x200488bc <R_SPI_Init>
163                                    R_LCD_Init();
20042b68: 0x31faffeb main+128        bl       0x20041434 <R_LCD_Init>
166                                    R_LED_Init();
20042b6c: 0x1effffeb main+132                 0x200427ec <R_LED_Init>
169                                  it_switches();
20042b70: 0x971200eb main+136        bl       0x200475d4 <init_switches>
172                                    display_image (pmptr, 128, 24, 0, 104);
20042b         0e3 main+140          mov      r3, #104         ; 0x68
20042b         de5 main+144          str      r3, [sp]
20042b7c: 0x10001be5 main+148        ldr      r0, [r11, #-16]
20042b80: 0x8010a0e3 main+152        mov      r1, #128         ; 0x80
20042b84: 0x1820a0e3 main+156        mov      r2, #24
20042b88: 0x0030a0e3 main+160        mov      r3, #0
20042b8c: 0x93faffeb main+164        bl       0x200415e0 <display_image>
173                                    display_on();
```

Figure 5-21. Disassembly View

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

1. Click **Window → Show View → Disassembly** or the icon to open the **Disassembly** view.
2. Click the icon to enable synchronization between the assembly source and the C source (active debug context).
3. In the **Disassembly** view, right-click on the address column to select **Show Opcodes** and **Show Function Offsets**.
4. You can enable and disable source addresses within the editor using the context menu.


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

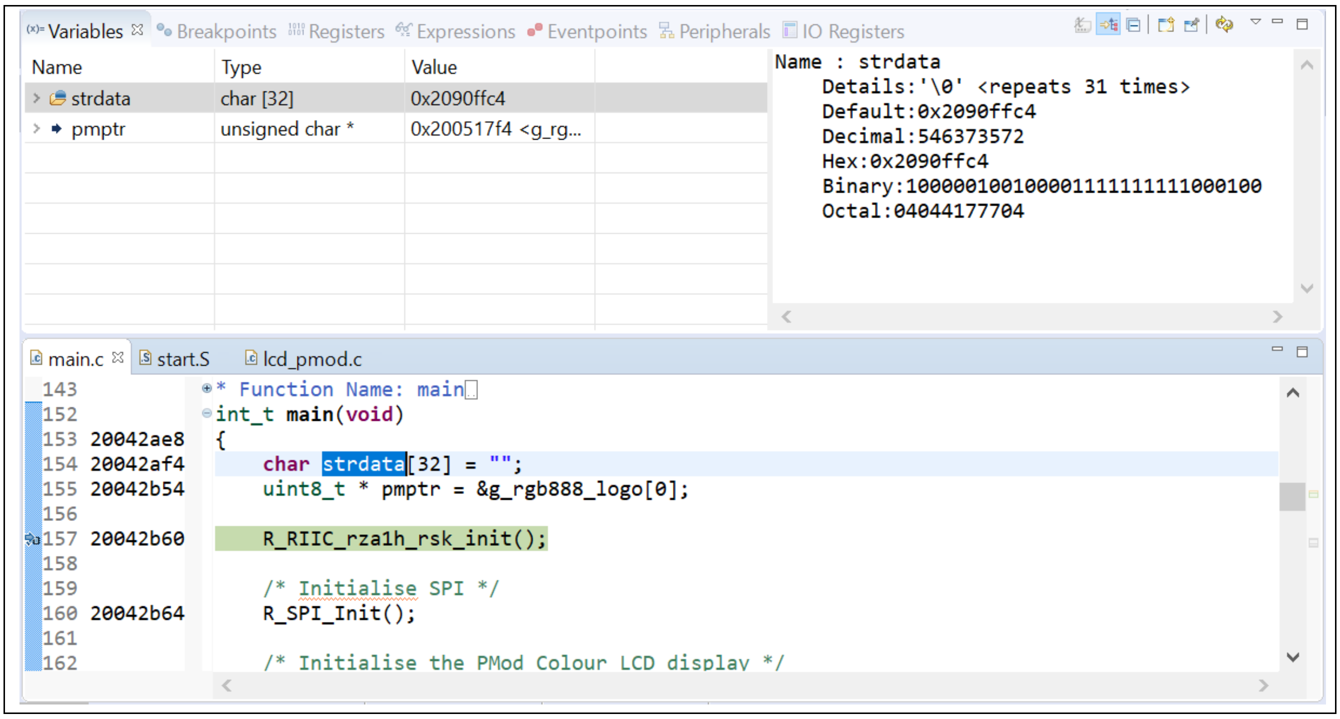
```
               /* End of function delay
               /* Function Name: main
               int_t main(void)
  20042b60     {
  20042b6c         char strdata[32] = "";
  20042b98         uint8_t * pmptr = &g_rgb888_logo[0];

  20042ba4         R_RIIC_rza1h_rsk_init();

                   /* Initialise SPI */
  20042ba8         R_SPI_Init();

                   /* Initialise the PMod Colour LCD display */
  20042bac         R_LCD_Init();

                   /* Initialise direct connected LED */
  20042bb0         R_LED_Init();
```

Figure 5-22. Source Addresses Menu

5.4.6 Variables View

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

Please refer to the **Expressions View** section to watch global variables or external variables out of the current program scope.



Name	Type	Value
strdata	char [32]	0x2090ffc4
pmptr	unsigned char *	0x200517f4 <g_rg...

```
Name : strdata
    Details:'\0' <repeats 31 times>
    Default:0x2090ffc4
    Decimal:546373572
    Hex:0x2090ffc4
    Binary:100000100100001111111111000100
    Octal:04044177704
```

```
                 * Function Name: main
                int_t main(void)
         20042ae8 {
         20042af4     char strdata[32] = "";
         20042b54     uint8_t * pmptr = &g_rgb888_logo[0];

         20042b60     R_RIIC_rza1h_rsk_init();

                      /* Initialise SPI */
         20042b64     R_SPI_Init();

                      /* Initialise the PMod Colour LCD display */
```

Figure 5-23. Variables View

To observe a local variable (e.g. `strdata` in function `main()`):

1. Click **Window** → **Show View** → **Variables** or the (x)= icon to open the **Variables** view.
2. Step into the function `main()` to view the value of local variable `strdata`.

Note:
Variables which are optimized out or temporarily allocated to accumulator registers may not appear in this view. Please refer to the **Disassembly** view if necessary.

5.4.7 Eventpoints View

An event refers to a combination of conditions set for executing break or trace features during program execution. The **Eventpoints** view allows you to set up or view defined events from various categories e.g. trace start, trace stop, trace record, event break, before PC, performance (timer) start and performance (timer) stop.

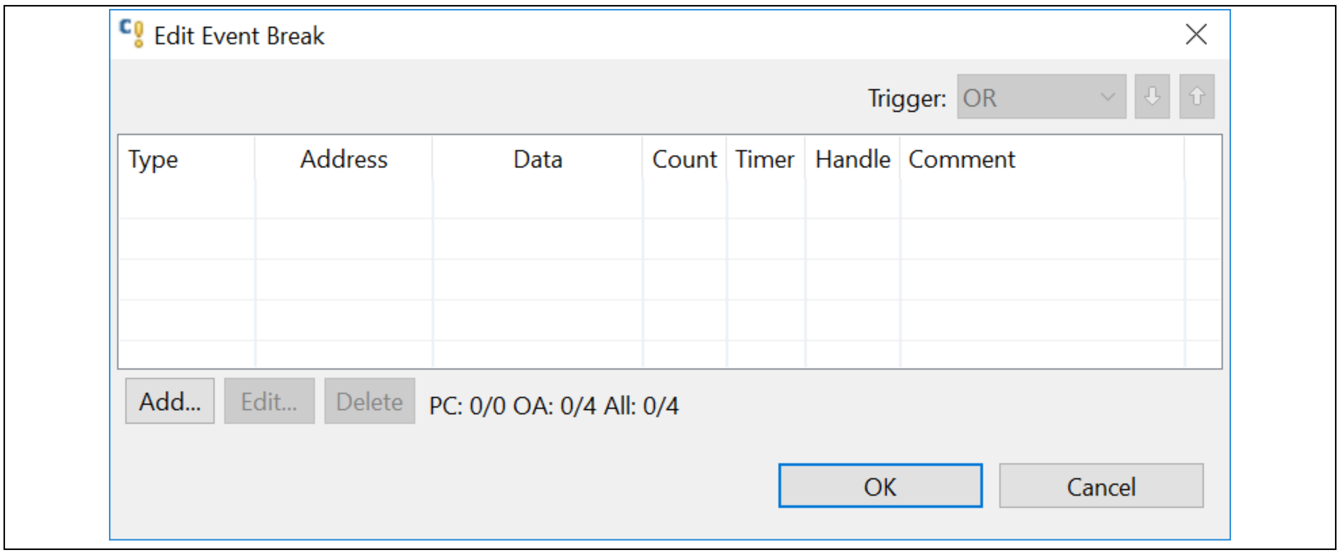
The number of events that can be set and the setting conditions differ with each MCU. There are two types of event:

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

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

To set an event break for a global variable when a variable is accessed (e.g. when `g_adc_result` is assigned a new value):

1. Click **Window → Show View → Eventpoints** or the icon to open the **Eventpoints** view.
2. Double-click the **Event Break** option to open **Edit Event Break** dialog box.
3. Click the **Add…** button to continue.

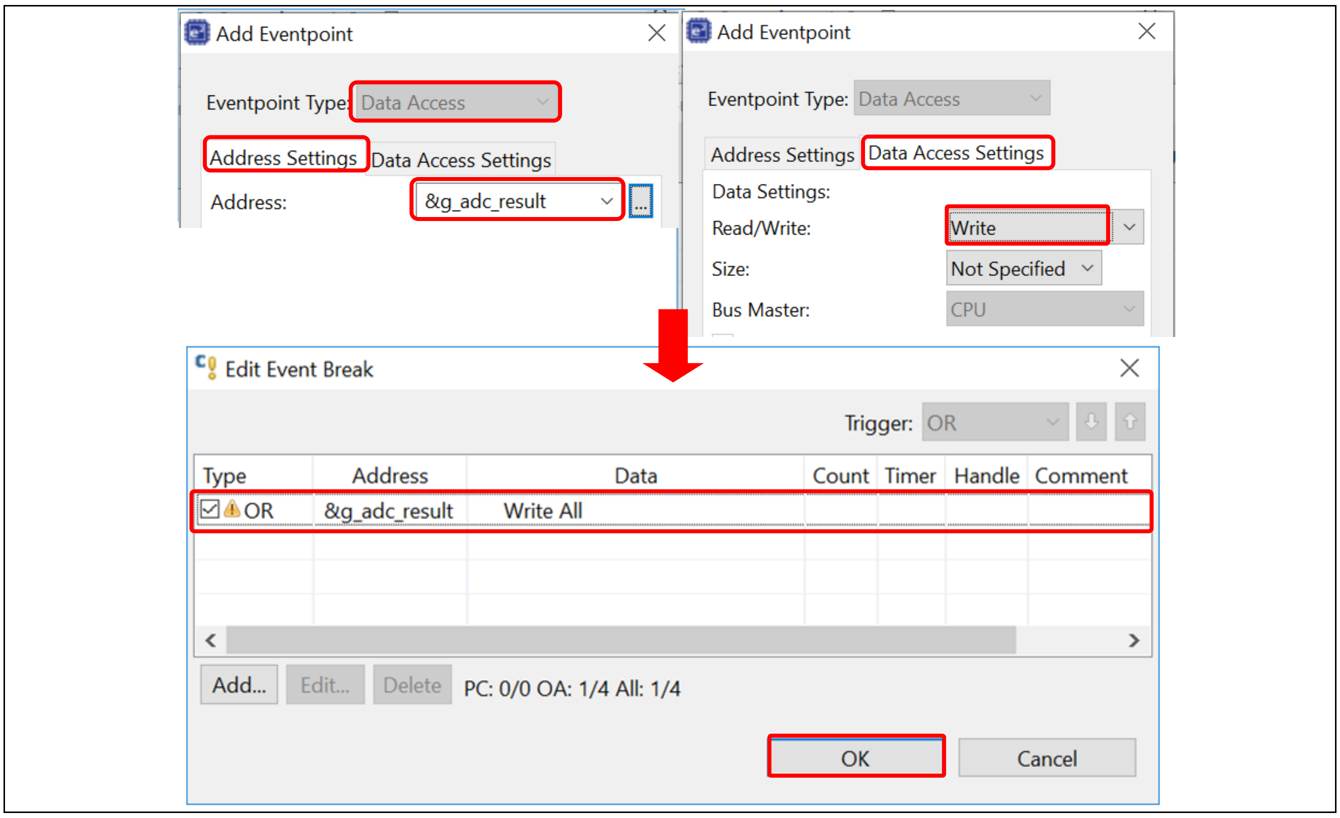


PC: 0/0 OA: 0/4 All: 0/4

Figure 5-24. Edit Event Break

4. Select **Data Access** as the eventpoint type.
5. Go to the **Address Settings** tab, click the icon and browse for the symbol `g_adc_result` (the address of this global variable is `&g_adc_result`).

6. Next, switch to the **Data Access Settings** tab, select 'Write' for **Data Settings**. Click **OK** to proceed.



Add Eventpoint

Eventpoint Type: Data Access

Address Settings | Data Access Settings

Address: &g_adc_result

Add Eventpoint

Eventpoint Type: Data Access

Address Settings | Data Access Settings

Data Settings:

Read/Write: Write

Size: Not Specified

Bus Master: CPU

Edit Event Break

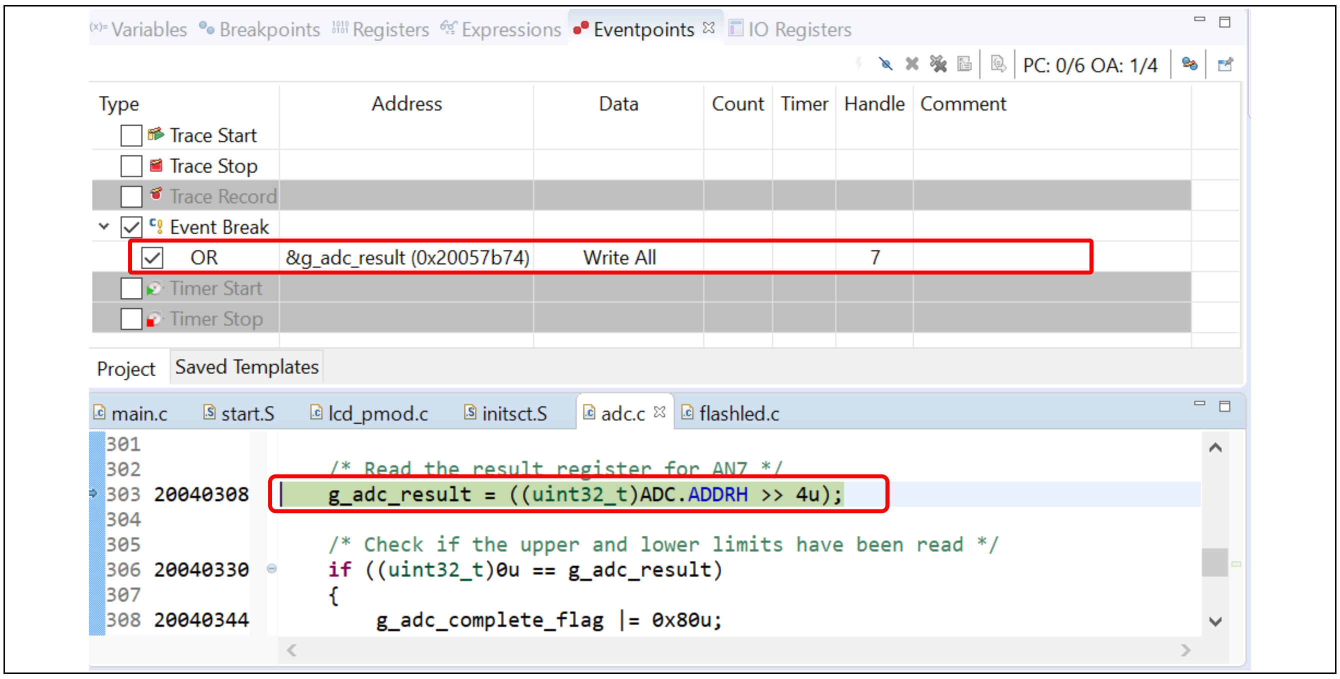
Trigger: OR

Type	Address	Data	Count	Timer	Handle	Comment
☑ OR	&g_adc_result	Write All				

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

Figure 5-25. Add Eventpoint

7. Ensure that the event break for `&g_adc_result` is set and enabled in the **Eventpoints** view. Reset to execute the program from the start.



Type	Address	Data	Count	Timer	Handle	Comment
☐ Trace Start						
☐ Trace Stop						
☐ Trace Record						
☑ Event Break						
☑ OR	&g_adc_result (0x20057b74)	Write All			7	
☐ Timer Start						
☐ Timer Stop						

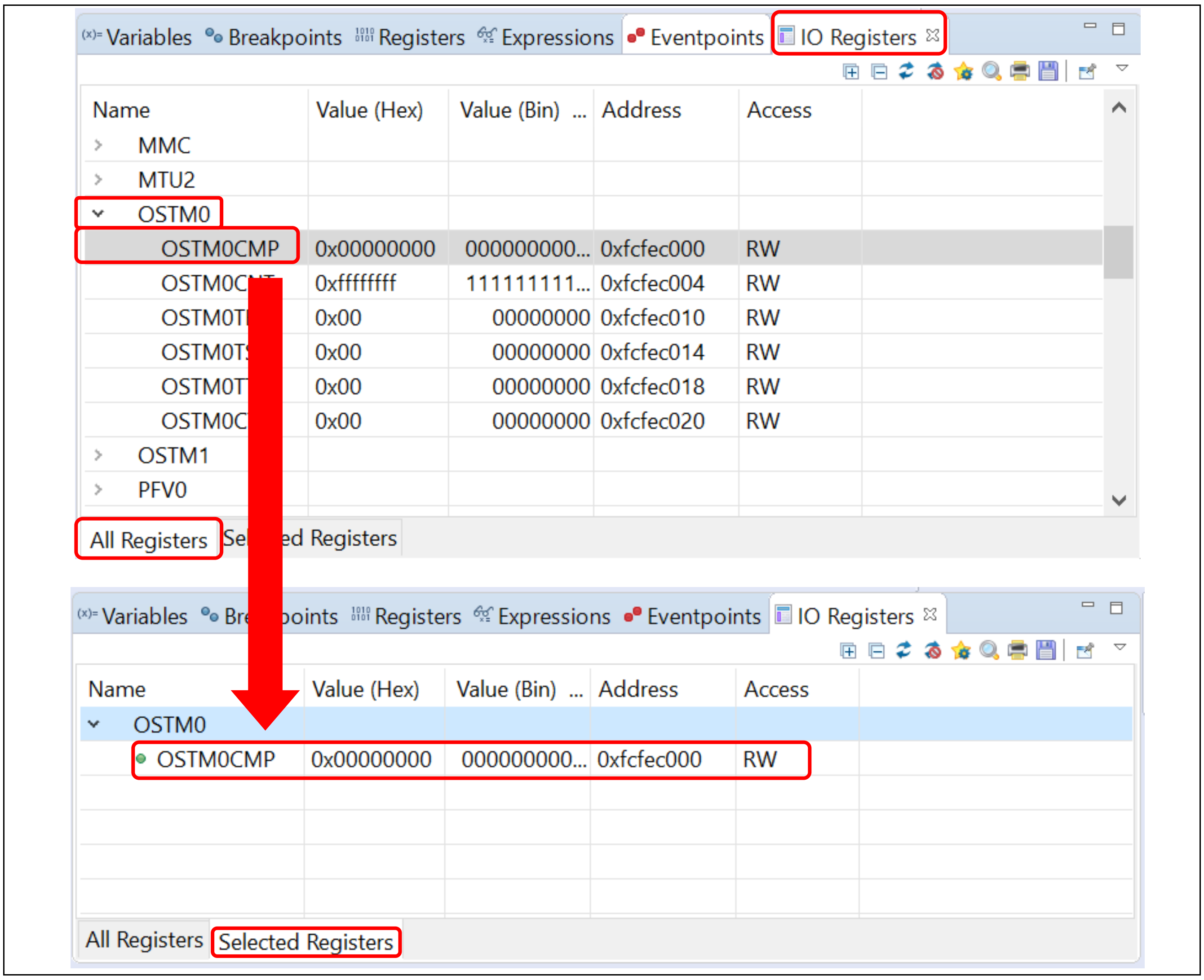
```
301
302                /* Read the result register for AN7 */
303  20040308      g_adc_result = ((uint32_t)ADC.ADDRH >> 4u);
304
305                /* Check if the upper and lower limits have been read */
306  20040330      if ((uint32_t)0u == g_adc_result)
307                {
308  20040344          g_adc_complete_flag |= 0x80u;
```

Figure 5-26. Execution of Event Break

Figure 5-26 shows that when `g_adc_result` is assigned a new value, the program stops at code line number 303 of `adc.c` (where `g_adc_result` is assigned new value).

5.4.8 IO Registers View

IO Registers is also known as the Special Function Registers (SFR). The **IO Register** view displays all the registers set defined in a target-specific I/O file, including their address, hexadecimal and binary values. You can further customize your own **IO registers** view by selectively adding I/O registers to the **Selected Registers** pane.



Name	Value (Hex)	Value (Bin) ...	Address	Access
MMC				
MTU2				
OSTM0				
OSTM0CMP	0x00000000	000000000...	0xfcfec000	RW
OSTM0C	0xffffffff	111111111...	0xfcfec004	RW
OSTM0T	0x00	00000000	0xfcfec010	RW
OSTM0T	0x00	00000000	0xfcfec014	RW
OSTM0T	0x00	00000000	0xfcfec018	RW
OSTM0C	0x00	00000000	0xfcfec020	RW
OSTM1				
PFV0				

Name	Value (Hex)	Value (Bin) ...	Address	Access
OSTM0				
● OSTM0CMP	0x00000000	000000000...	0xfcfec000	RW

Figure 5-27. IO Registers View

To view selected I/O registers (e.g. OSTM0CMP in OSTM0):

1. Click **Windows → Show View → Others…** In the 'Show View' dialog, click **IO Registers** under **Debug** or the icon to open the **IO Registers** view.
2. Under the **All Registers** tab, locate **OSTM0** in the **IO Registers** view. Expand the OSTM0 I/O register list.
3. Drag and drop the 'OSTM0CMP' to the **Selected Registers** pane. A green dot ● besides the I/O register indicates the status of being the selected register(s).
4. Switch to the **Selected Registers** tab to view 'OSTM0CMP' of the 'OSTM0' I/O register.

The expanded I/O register list may take a longer time to load in the **All Registers** pane. Hence, it is advisable to customize and view multiple selected I/O 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 is helpful when tracking the program execution flow to search for and examine points where problems have arisen.

The trace buffer is limited to a size of 1 to 32 Mbytes. The oldest trace data is overwritten with the new data once the buffer has become full.

Tracing is not fully supported on the RZ/A1H.

5.5 How to debug (RZ/A2M)

This chapter describes how to debug referring to the 'Loader' and 'Application' projects built in Chapter 3.2.1 and based on the hardware configuration: Segger J-Link emulator and RZ/A2M Evaluation Board Kit as target board.

1. Click the 'Loader' project in the **Project Explorer** pane to set focus.
2. Build the 'Loader Project' (refer to Chapter 4.2), to create the 'Loader.elf' file.
3. Select 'Loader HardwareDebug' in the **Launch Configuration** of the Launch Bar.

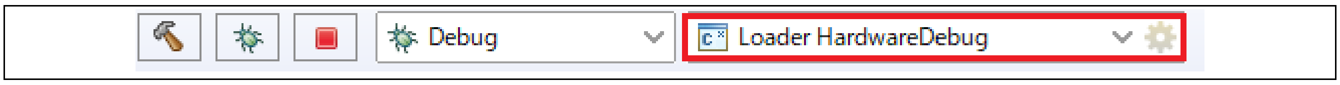
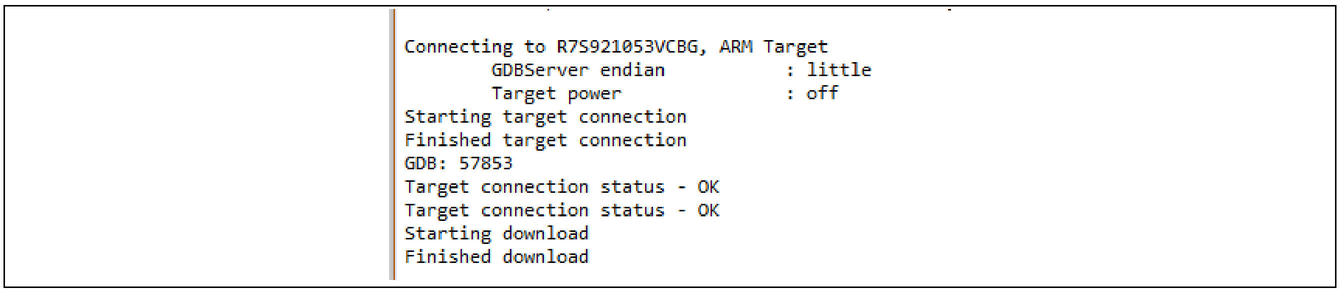


Figure 5-28. Select Launch Configuration

4. Click the button of Launch Bar, and connect to target board and download 'Loader.elf' file to it.



```
Connecting to R7S921053VCBG, ARM Target
        GDBServer endian            : little
        Target power                : off
Starting target connection
Finished target connection
GDB: 57853
Target connection status - OK
Target connection status - OK
Starting download
Finished download
```

Figure 5-29. Console View After Download

5. Click the button on the Launch Bar to stop the debug session.
6. Click the 'Application' project in **Project Explorer** pane to set focus.
7. Build the 'Application Project' (refer to Chapter 4.2), to create the 'Application.elf' file.
8. Select 'Loader HardwareDebug' in the **Launch Configuration** of the Launch Bar.

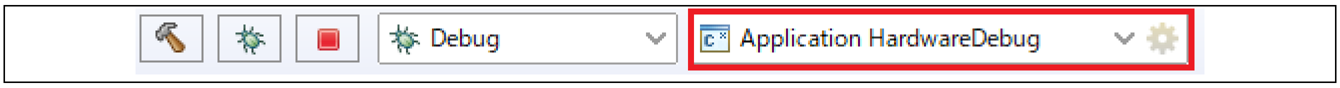


Figure 5-30. Select Launch Configuration

9. Click the button on the Launch Bar, to connect to the target board and download the 'Application.elf' file to it.
10. After downloading the 'Application.elf' file, click the button to restart the program from the entry point of 'Loader.elf'.

6. Help

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

Click on the **Help** tab to pull down the help menu.

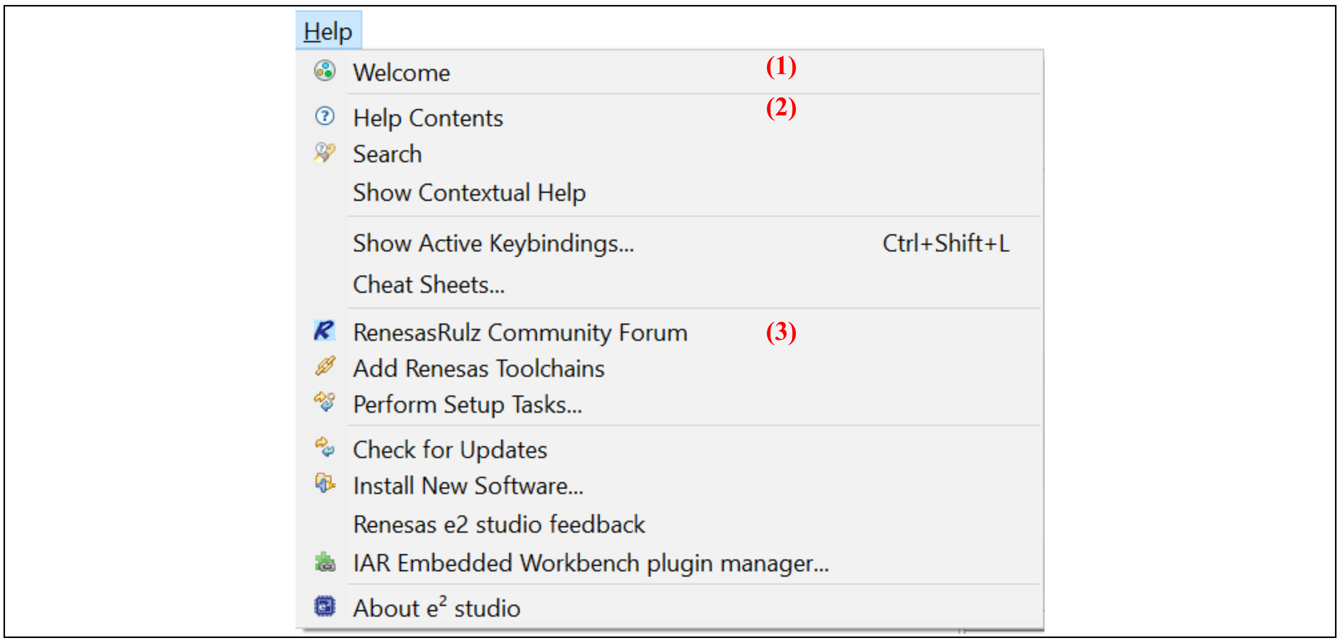


Figure 6-1. Help Menu

Quick Help Tips

(1) Click **Welcome** for an overview of e^2 studio, a link to access the IDE tutorial and sample, and to view the release notes.

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

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

Revision History	e^2 studio Integrated Development Environment User's Manual: Getting Started Guide

Rev.	Date	Description	
		Page	Summary
1.00	Jun.12.19	—	First Edition issued
1.01	Mar. 3.20	19-30	Add 'Project Generation of RZ/A2M'
		64	Add 'How to debug (RZ/A2M)'
1.02	Sep. 24.20	—	Updated for e^2 studio 7.8.0 and 2020-04 support

e^2 studio Integrated Development Environment
User's Manual: Getting Started Guide

Publication Date: Rev.1.02 Sep 24, 2020

Published by: Renesas Electronics Corporation

e^2 studio

Renesas Electronics Corporation

R20UT4535EJ0102