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

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.

2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.

3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.

4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.

5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.

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

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implants; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.

6. When using Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.

7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.

8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.

9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.

10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.

11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.

12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.
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 VIL (Max.) and VIH (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 VIL (Max.) and VIH (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 e² studio integrated development environment for developing applications and systems and provides an outline of its features.

e² 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 e² studio and who design software and hardware applications.

Purpose
This manual aims to provide the user with the explanation of the features provided by e² 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
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>ARM</td>
<td>Advanced RISC Machine</td>
</tr>
<tr>
<td>CAN</td>
<td>Controller Area Network</td>
</tr>
<tr>
<td>CDT</td>
<td>C/C++ Development Tooling</td>
</tr>
<tr>
<td>CG</td>
<td>peripheral Code Generator</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>ELF/DWARF</td>
<td>Executable and Linkable Format / Debugging With Attributed Record Formats</td>
</tr>
<tr>
<td>FIT</td>
<td>Firmware Integration Technology</td>
</tr>
<tr>
<td>GB</td>
<td>Gigabyte</td>
</tr>
<tr>
<td>GCC</td>
<td>GNU Compiler Collection</td>
</tr>
<tr>
<td>GDB</td>
<td>GNU project DeBugger</td>
</tr>
<tr>
<td>GHz</td>
<td>Giga Hertz</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>I/O</td>
<td>Input / Output</td>
</tr>
<tr>
<td>JTAG</td>
<td>Joint Test Action Group</td>
</tr>
<tr>
<td>LCD</td>
<td>Liquid Crystal Display</td>
</tr>
<tr>
<td>LED</td>
<td>Light Emitting Diode</td>
</tr>
<tr>
<td>LVDS</td>
<td>Low-Voltage Differential Signalling</td>
</tr>
<tr>
<td>MCU</td>
<td>Microcontroller Unit</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer or Program Counter</td>
</tr>
<tr>
<td>PMOD</td>
<td>Peripheral MODule (an open standard defined by Digilent Inc™)</td>
</tr>
<tr>
<td>RTOS</td>
<td>Real Time Operating System</td>
</tr>
<tr>
<td>SC</td>
<td>Smart Configurator</td>
</tr>
<tr>
<td>SD</td>
<td>Secure Digital</td>
</tr>
<tr>
<td>SFR</td>
<td>Special Function Registers</td>
</tr>
<tr>
<td>SVN</td>
<td>SubVersioN</td>
</tr>
<tr>
<td>TFT</td>
<td>Thin Film Transistor</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>USB</td>
<td>Universal Serial Bus</td>
</tr>
</tbody>
</table>
# Table of Contents

1. General .............................................................................................................................. 1
   1.1 System Configuration................................................................................................................................. 1
   1.2 System Requirements ................................................................................................................................ 1
   1.3 Supported Compiler ................................................................................................................................... 2
   1.4 Additional Tool for Compiler Update .......................................................................................................... 2
   1.5 Supported Emulator ................................................................................................................................... 2
   1.6 Supported Simulator ................................................................................................................................. 2

2. Installation ......................................................................................................................... 3
   2.1 Installation of e² studio IDE ........................................................................................................................ 3
   2.2 Installation of Additional Software .............................................................................................................. 7
   2.3 Uninstallation of the e² studio IDE ............................................................................................................. 10
   2.4 Major Version Upgrade for the e² studio IDE ........................................................................................... 10
   2.5 Update plugins over Internet .................................................................................................................... 10
   2.6 Register Toolchain to e² studio ................................................................................................................ 14

3. Import / Generate Project ................................................................................................ 16
   3.1 Import Existing Project ............................................................................................................................. 16
   3.2 New Project Generation ........................................................................................................................... 19

4. Build .............................................................................................................................................. 36
   4.1 Build Option Settings ............................................................................................................................... 36
   4.2 Building the Sample Project ..................................................................................................................... 38
   4.3 Exporting Build Configuration Settings .................................................................................................... 39

5. Debug ............................................................................................................................................ 40
   5.1 Change Existing Debug Configurations ................................................................................................... 40
   5.2 Create New Debug Configurations .......................................................................................................... 44
   5.3 Launch Bar ............................................................................................................................................... 45
   5.4 Basic Debugging Features ....................................................................................................................... 46
   5.5 How to debug (RZ/A2M) .......................................................................................................................... 64

6. Help .............................................................................................................................................. 65
1. General

Renesas eclipse embedded studio (known as ‘e2 studio’) is a complete, state of the art development environment supporting Renesas embedded micro-controllers. It is developed based on the popular open-source Eclipse CDT (C/C++ Development Tooling) project that covers the build (editor, compiler and linker control) and debug phases with an extended GDB (GNU Project Debugger) interface support.

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

![System Configuration Example - RZ/A1H Renesas Starter Kit](image)

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 1GB (2GB or larger is recommended, especially for Windows 64-bit OS)
- Display: Resolution at least 1,024 x 768; at least 65,536 colors
- Interface: USB 2.0 (High-speed/Full-speed). High-speed is recommended.

Software Environment:

- Windows® 10 (32- and 64-bit versions)
- Windows® 8.1 (32- and 64-bit versions)
- Windows® 7 (32- and 64-bit versions)
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 e2 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 e2 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 e2 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

1.6 Supported Simulator

None
2. Installation

The latest e² studio IDE installer package can be downloaded from the Renesas website for free. You will need to login to your Renesas account (on the MyRenesas page) to download the software.

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

**Note 1:** e² studio does not support the update of V5.4 or earlier versions to V6.0.0 or later. Please uninstall the earlier versions before installation. Alternatively, install V6.0 or later into a new folder.

**Note 2:** e² studio V6 installer has a ‘Modify’ function to add or remove features to the existing installation. Please update e² studio with Help → Check for Updates before using the ‘Modify’ function.

2.1 Installation of e² studio IDE

1. Download the e² studio installer version 7.0 or higher from the official Renesas Electronics Corp: https://www.renesas.com/e2_studio

2. Double-click on the e² studio installer to launch the e² studio installation wizard. Click the **Next** button to continue.

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.

   **Note:** Multi-byte characters cannot be used for the e² studio installation folder name.

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

![Figure 2-1. Device Family Pane: Support for RZ Devices](image)
Note: **Support for RZ/G Devices** is the option for RZ/G Linux application project generation and debugging. If you need to develop applications for Linux on RZ/G device, please select this too.

![Figure 2-2. Device Family Pane: Support for RZ/G Devices](image)

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.

![Figure 2-3. Extra Components Pane: RTOS is Selected](image)
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.

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 e² studio installer (for more information, see chapter 2.2). 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.
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.

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

10. Summary
    Click the **Install** button to install the Renesas e² studio IDE.

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.
2.2 Installation of Additional Software

The e² 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).

2.2.1 GNU ARM Embedded Toolchain

If it was selected in the Additional Software pane of e² 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.

![Image of installation wizard steps]

Figure 2-6. Installation of GNU ARM Embedded Toolchain
2.2.2 Libgen Update for GNU ARM Embedded Toolchains

If it was selected in the Additional Software pane of e² 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.

![Figure 2-7. Libgen Update for GNU ARM Embedded: Select Toolchain Folder](image)
3. Next, click **Install Libgen Update for GNU ARM Embedded Toolchains.**

![Image of Libgen Update for GNU ARM Embedded Toolchains]

**Figure 2-8. 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).
2.3 Uninstallation of the e² studio IDE

The e² 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 e² studio.

At the end of the uninstallation, the e² studio IDE will be deleted from the installed location and Windows shortcut menus will be removed.

2.4 Major Version Upgrade for the e² studio IDE

The **Check for Updates** operation does not work for major version upgrades (a major digit version number increase, for example from version 6.3.0 to version 7.0.0), you need to use the e² 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.

2.5 Update plugins over Internet

The e² studio IDE supports online and offline updates for minor version updates (i.e. minor digit number increases, for example going from version 7.3.0 to version 7.4.0). This allows the available software components to be updated over the internet or without an internet connection.

2.5.1 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.
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.

![Figure 2-10. e² studio – Available Updates Panel (1/3)](image)

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

![Figure 2-11. e² studio – Available Updates Panel (2/3)](image)
4. Read and accept the software license agreement. Click the **Finish** button to complete the update.

![Figure 2-12. e² studio – Available Updates Panel (3/3)](image)

5. Click the **Help → About e² studio** menu option to check the version number and verify the update.

![Figure 2-13. e² studio – About e² studio Panel](image)
2.5.2 Offline Minor Version Update

This chapter shows how to update e² 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 e² 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 e² studio to a different folder.

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

![Figure 2-14. Upgrade e² studio from Offline Installer](image)

4. Follow the steps shown in Chapter 2.1 Installation of e² studio IDE. Step 2 Install Folder will be skipped since the upgrade option will use the same destination folder as the existing e² studio installation.
2.6 Register Toolchain to e² studio

The e² studio installer can install toolchains (such as GCC ARM Embedded) automatically during e² studio installation with a valid Internet connection. However, in situations where an Internet connection is not available during e² studio installation, toolchains that are needed can be installed later by using installer files from the web site and then registered to e² studio. For the toolchain download site for RZ, please refer to chapter 1.3.

e² studio scans for installed toolchains on start up. After a new toolchain installation, e² studio will pop up a toolchain integration window to allow for toolchain selection.

Figure 2-15. e² 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 e² studio.

Figure 2-16. e² studio Add Renesas Toolchains
If the desired toolchain is not listed, Click **Add…** and specify the location where it is installed.

Figure 2-17. 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 right-click inside **Project Explorer** or select the menu **File → Import**

![Figure 3-1. Import Menu](image)
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.

![Figure 3-2 Select Folder or Archive File Included Project Files](image1)

5. Projects generated by legacy e² studio (before V6.0) will need to be upgraded. Click the **Project Upgrade Required** pop-up message (continued on the next page).

![Figure 3-3. Project Upgrade](image2)
Alternatively, run **Upgrade Legacy e2 studio Projects**... from the project right-button context menu.

![Figure 3-4. Manual Project Upgrade](image)

Click **Yes** on the **Migration Required** dialog box to confirm the migration (Figure 3-5).

![Figure 3-5. Migration Required Dialog Box](image)
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 type of RZ/A2M project 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 e² 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.

![Image of 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.

![Figure 3-7. Select Project Template](image)

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

![Figure 3-8. Specify Project Name](image)
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.

![Figure 3-9. Select Toolchain, Device & Debug Settings: e.g. RZ/A2M Renesas Starter Kit](image)

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

![Figure 3-10. Select Coding Assistant Settings](image)
6. In the initial settings, USB and RTC are not used. In ‘Loader Project’, Click **Next** to proceed.

![Figure 3-11. Select Peripheral Usage Options](image)

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

![Figure 3-12. Select Additional CPU Options](image)
8. Select ‘Project-Built’ and click **Next** to proceed.

![Figure 3-13. Library Generator Settings: Select Project Built](image)

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

![Figure 3-14. New Project Creation Wizard (Summary)](image)
10. A new C project named ‘Loader’ will be created.

![Figure 3-15. New C Project Created](image)

11. Next, create an Application project.

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

![Figure 3-16 Open New Project Creation Wizard](image)
12. Select the template for the new project (for example, Renesas RZ: ‘GCC for Renesas RZ C/C++ Executable Project’). Click **Next** to proceed.

![Figure 3-17. Select Project Template](image)

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

![Figure 3-18. Specify Project Name](image)
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.

![Select Toolchain, Device & Debug Settings](image)

*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.

![Image of Coding Assistant Settings]

**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.

![Image of Peripheral Usage Options]

**Figure 3-21. Select Peripheral Usage Options**
17. Keep the default settings for additional CPU options and click **Next** to proceed.

![Image](image-url)  
**Figure 3-22. Select Additional CPU Options**

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

![Image](image-url)  
**Figure 3-23. Select Library Generator Settings**
19. A project summary is displayed. Click **Finish** to generate the project.

![Figure 3-24. New Project Creation Wizard (Summary)](image)

20. Click **Open Perspective**.

![Figure 3-25. Open Associated Perspective Wizard](image)
21. A new C project named ‘Application’ is 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, e² studio supports RZ Linux C/C++ project for developing Linux application. The Support for RZ/G option has to be selected during e² studio installation. refer to Figure 2-2.

To create a new project, invoke e² studio IDE from the Windows Start menu and specify a workspace directory.

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

![Figure 3-27. Open New Project Creation Wizard](image)

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

![Figure 3-28. Select Project Template](image)
3. Enter the project name. Click **Next** to proceed.

![Figure 3-29. Specify Project Name](image)

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.

![Figure 3-30. Select toolchain, device & debug settings: e.g. RZ/A1H Renesas Starter Kit](image)
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.

![Figure 3-31. Select Coding Assistant Settings](image)

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

![Figure 3-32. Select Additional CPU Options](image)
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.

![Figure 3-33. Library Generator Settings Warning Message](image)

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

![Figure 3-34. New Project Creation Wizard (Summary)](image)
10. A new C project named 'Tutorial' will be created.

![Figure 3-35. New C Project Created](image)

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 e² 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](image)
2. Click **C/C++ Build → Settings → Toolchain** to view or change the toolchain version.

![Change Toolchain Version](image1)

**Figure 4-2. Change Toolchain Version**

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

![Build Settings for Compiler: Environment](image2)

**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](image)

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.

![Figure 4-5. Project Has Built Successfully](image)

### 4.3 Exporting Build Configuration Settings

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

1. Right-click 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](image)
5. Debug

This chapter describes the operation of the debug configuration and key debugging features of e² 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 e² studio IDE and select the **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 ‘e² studio Debug Help’ as described in chapter 6.

5.1 Change Existing Debug Configurations

The debug configuration has to be configured when debugging for the first time and 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.
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**.

![Figure 5-3. Select Load Module](image)

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

![Figure 5-4. Select Target Device](image)
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.

![Figure 5-5. Change Connection Settings](image-url)
6. Switch to the **Debug Tool Settings** sub tab, based on the RSK RZ/A1H board to ensure:

- Memory
  - Endian = ‘Little Endian’

![Image of Debug Tool Settings]

**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`.

![Image of Debug View]

**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.

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

![Duplicate A Selected Debug Launch Configuration](image)
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.

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 e² studio’s main window. This simple interface includes build and debug buttons for the selected launch target.

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 e² studio IDE.

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

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

![Useful Toolbars in Debug Views](image)

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


2. In e² studio, select File → Import

![Figure 5-12. Importing the Sample Project](image)

3. In the Import dialog, select General → Existing Projects into Workspace. Click Next

![Figure 5-13. The Select Import Wizard](image)
4. In the **Import Projects** dialog, select **Select archive file**. Click **Browse** then select the downloaded archive file.

![Figure 5-14. Select project location to import](image)

5. Select project ‘RZ_A1H_Tutorial_RSK’ from the **Projects** list, then click **Finish**

![Figure 5-15. Selecting the Sample Project](image)
6. Right click on the imported project and select **Upgrade Legacy e2 studio Projects**...

![Figure 5-16. Upgrading the Imported Project](image)

7. Select the ‘RZ_A1H_Tutorial_RSK’ project and then click **Finish**

![Figure 5-17. Finishing the Upgrade](image)
8. 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.

![Figure 5-18. Update Project Toolchain](image)

9. Switch to the Device tab to check if the current device matches with the device on the board. If not, click Change Device and select the correct device.

![Figure 5-19. Change Device](image)
10. After selecting the new device, click **generate** to regenerate the files relating to the new device.

![Figure 5-20. Regenerate Files After Changing Device](image)

11. 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**.

![Figure 5-21. Add Compiler Generated Script File](image)

12. 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.

![Figure 5-22. Breakpoints view](image)
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.

![Figure 5-23. Expressions View](image)

**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.

![Figure 5-24. Registers View](image)

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.

![Figure 5-25. Memory View (1/2)](image)

The global variable g_data_block_content is presented in memory renderings of ‘Hex Integer’ format.
To add new renderings format (e.g. ASCII) for the variable \texttt{g\_data\_block\_content}:

1. Click the \textit{New Renderings...} tab and select ‘ASCII’ to add the rendering.
   
   This creates a new tab named ‘\texttt{&g\_data\_block\_content < ASCII}’ next to the tab ‘\texttt{&g\_data\_block\_content <Hex Integer}>’.

\textbf{Figure 5-26. 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.

![Disassembly View](image)

**Figure 5-27. Disassembly View**

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

![Figure 5-28. 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.

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

1. Click Window → Show View → Variables or the 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](image) 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.

4. Select *Data Access* as the eventpoint type.
5. Go to the *Address Settings* tab, click the ![icon](image) 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.

![Figure 5-31. Add Eventpoint](image)

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.

![Figure 5-32. Execution of Event Break](image)

Figure 5-33 shows that when \texttt{g\_adc\_result} is assigned a new value, the program stops at code line number 303 of \texttt{adc.c} (where \texttt{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.

Figure 5-33. 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.

![Figure 5-34. Select Launch Configuration](image)

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

![Figure 5-35. Console View After Download](image)

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 5-36. Select Launch Configuration](image)

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 e² studio from here.

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

![Figure 6-1. Help Menu](image)

**Quick Help Tips**

1. Click **Welcome** for an overview of e² 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 e² studio IDE. Internet connection is required.
## Revision History

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Page</th>
<th>Description</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>Jun.12.19</td>
<td>—</td>
<td>First Edition issued</td>
<td></td>
</tr>
<tr>
<td>1.01</td>
<td>Mar. 3.20</td>
<td>19-30</td>
<td>Add 'Project Generation of RZ/A2M'</td>
<td>Add 'How to debug (RZ/A2M)'</td>
</tr>
</tbody>
</table>