RZ/A1LU Group

Stream it! - RZ Tutorial Manual

Introduction

This application note uses the Stream it! - RZ V2 kit to run the Tutorial.

The hardware needed to follow this application note includes:

- Windows™ 7/ 8/ 8.1/ 10 compatible PC
- Stream it! - RZ V2 kit including display
- USB to micro USB Cable
- Segger J-Link Lite Debugger

The software components that will be obtained while following this application note include:

- e² studio (Recommended latest version)
- GNU ARM NONE Embedded Compiler (Version 16.01)

> This document refers to many third party website resources. These websites are not controlled by Renesas Electronics, and we are therefore unable to offer support for these resources.

The following documents apply to the RZ/A1LU based Renesas Stream it! - RZ V2. Please refer to the latest versions of these documents.

Document Type	Description	Document Title	Available from
Hardware Manual	Provides technical details of the RZ/A1LU microcontroller.	RZ/A1LU Group User's Manual: Hardware	https://www.renesas.com/en-eu/products/microcontrollers-microprocessors/rz/rza/rza1lu.html

Target Device

RZ/A1LU Group

Contents

1. Overview

This document aims to guide the user through the Tutorial for the Stream it! - RZ V2 product and provides an evaluation of the following features:

- Renesas microcontroller programming
- User code debugging
- User circuitry such as a switch, LED, and potentiometer
- Sample application
- Sample peripheral device initialisation code

1.1 Licenses

This sample application does not include any third party code applications.

1.2 Introduction

This manual is designed to answer, in tutorial form, the most common questions asked about using a Renesas Stream it! - RZ board. The tutorials help explain the following:

- How do I compile, link, download, and run a simple program on the Stream it! - RZ board?
- How do I build an embedded application?
- How do I use Renesas' tools?

Files referred to in this manual are installed using the import wizard as you work through the tutorials. The tutorial examples in this manual assume that installation procedures described in the Stream it! - RZ Quick Start Guide have been completed. Please refer to the Quick Start Guide for details of preparing the configuration.

These tutorials are designed to show you how to use the Stream it! - RZ and are not intended as a comprehensive introduction to the e^2 studio environment, compiler toolchains, or the J-Link LITE debugger. Please refer to the relevant user manuals for more in-depth information.

1.3 Note Regarding Source Code

Due to the project generator, it is possible that line numbers for source code illustrated in this document do not match exactly with that in the actual source files. It is also possible that the source address of instructions illustrated in this manual differ from those in user code compiled from the same source. These differences are minor, and do not affect the functionality of the sample code nor the validity of this manual.

2. Tutorial Project Workspace

2.1 Introduction

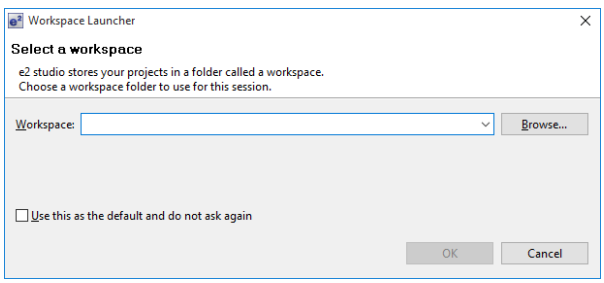
e^2 studio is an integrated development tool that allows the user to write, compile, program and debug a software project on the RZ family of Renesas microcontrollers. This application has been written as an e^2 studio workspace and therefore e^2 studio is required. For details on how to install e^2 studio see section 4.1 of this document.

This manual will describe the stages required to create and debug the supplied tutorial code.

2.2 Starting e^2 studio and Importing Sample Code

- Start e^2 studio
  Windows™ 7: Start Menu > All Programs > Renesas Electronics e2studio > e2 studio
  Windows™ 8 / 8.1: From Apps View, click 'e^2 studio' icon
  Windows™ 10: Start Menu > All apps > Renesas Electronics e2studio > e2 studio

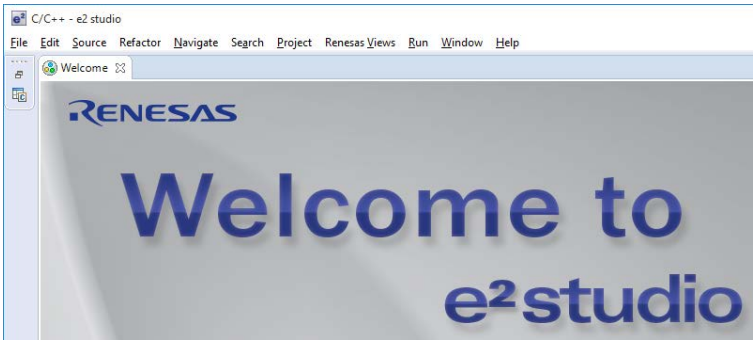
  The first dialog to appear will be the Workspace Launcher.



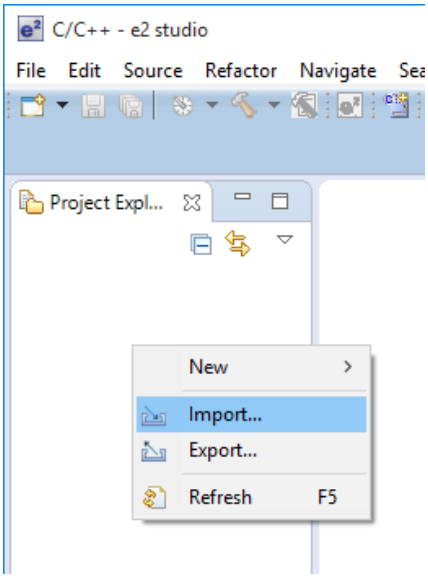
- Click ‘Browse’ and select a suitable location to store your workspace, using the ‘Create New Folder’ option as necessary. Click ‘OK’.

Note: The Workspace location does not have to contain your project files, the workspace contains the configuration of the tool and can group projects together. Projects may be referred to from this location, or the projects may be stored under this directory.

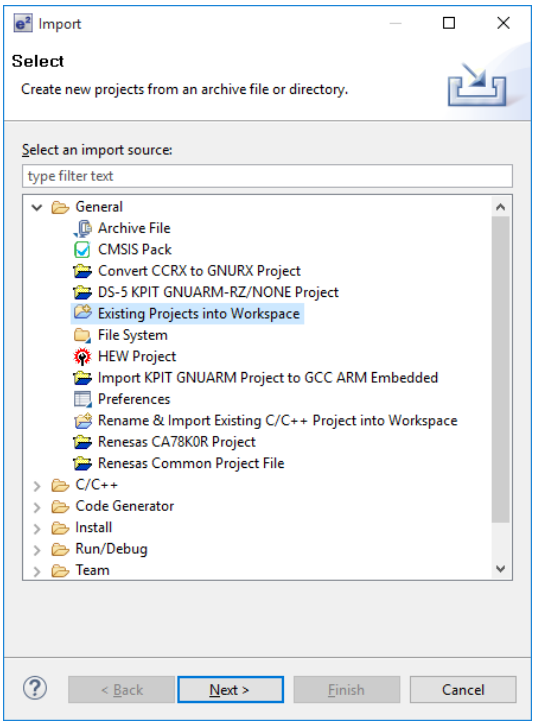
- e^2 studio will open with the ‘Welcome…’ tab as shown opposite.
- Close the tab by clicking on the cross.



- Right click in the ‘Project Explorer’ window and select ‘Import…’



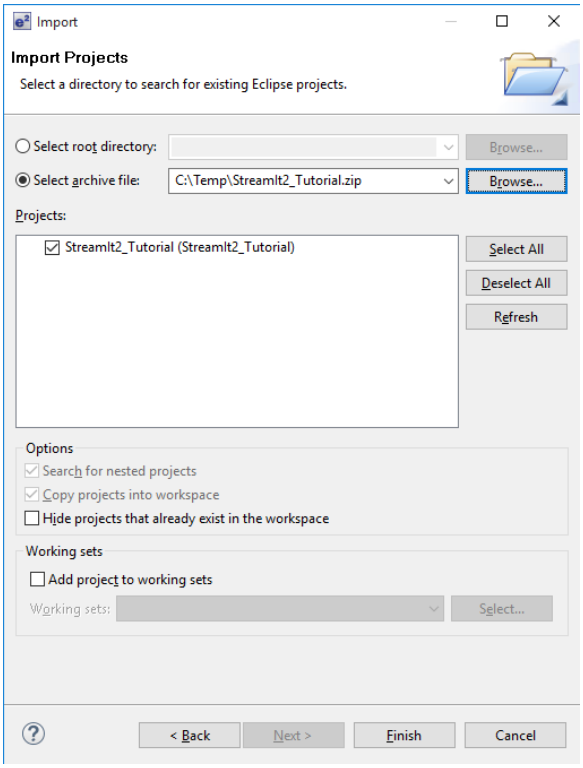
- The Import dialog box will now show. Expand the ‘General’ folder icon, and select ‘Existing Projects into Workspace’, then click ‘Next >’.



- The Import dialog box will allow you to specify a project to import. Click the ‘Browse’ button and locate the following directory:

  C:\Renesas\StreamIt2_Tutorial

- Press ‘OK’ and select tutorial project [StreamIt2_Tutorial]
- Ensure that the ‘Copy projects into workspace’ option is ticked.
- Caution: Ticking this box will copy the projects from the location where they were installed. It is important to select this option to preserve the projects that were installed so that you can return to them in the future.
- Click ‘Finish’.



2.3 Build Configurations and Debug Sessions

2.3.1. Build Configurations

The e^2 studio workspace will be created with two build configurations: ‘Release’ and ‘HardwareDebug’.

Release

This build mode has optimisation turned on, and provides little debug information. The C code execution may appear to be out of order, due to the way the compiler optimises the code. This build configuration is intended for final ROM-programmable code.

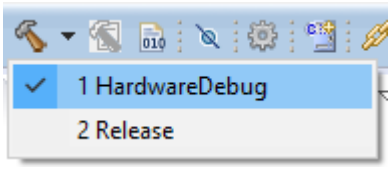
HardwareDebug

This default build mode has all optimisation turned off, and provides full debug information. This is the best configuration to use whilst developing programs as C code execution will be linear. The ‘HardwareDebug’ build configuration provided for this Tutorial program is configured to load the code directly into RAM.

2.3.2. Building the code

- Click the top level ‘Tutorial’ folder again, and then the arrow next to the build button  (hammer icon), and select the ‘HardwareDebug’ option.

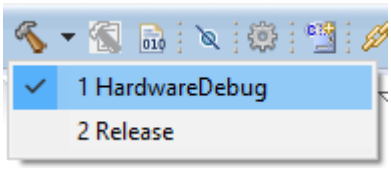
  e^2 studio will now build the code.



- The output from the build process will be presented in the console window of e^2 studio.

2.3.3. Debug Configuration

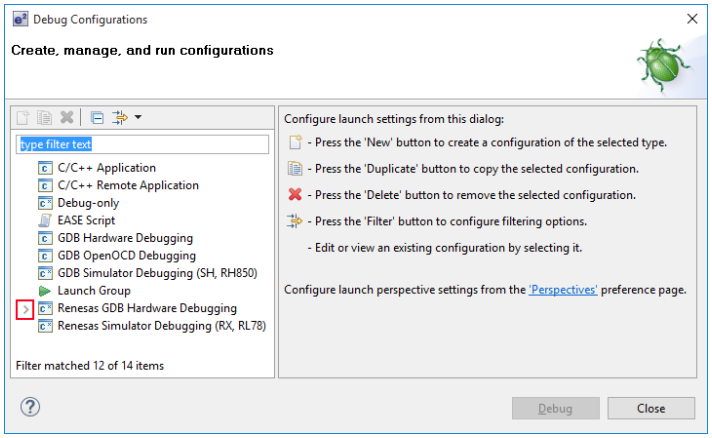
Click the arrow next to the debug button  (bug icon). Select ‘Debug Configurations…’.



- The ‘Debug Configurations’ dialog box will appear. Click on the arrow next to ‘Renesas GDB Hardware Debugging’ to expand the view.
- By default, e^2 studio creates Debug Configurations for each existing build mode. The Stream it! - RZ project has pre-configured debug configurations that are ready to use.

  Note:

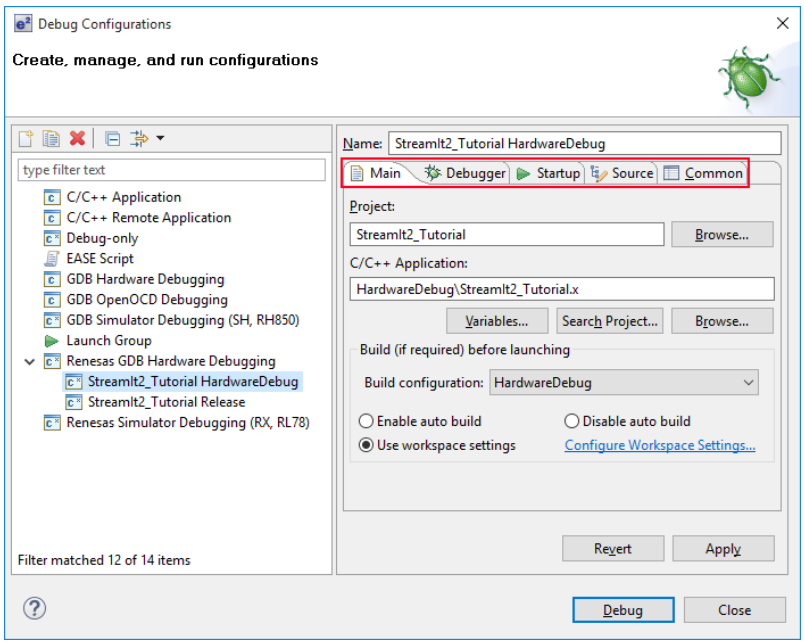
  To manually create a new e^2 studio debug configuration, click on ‘Renesas GDB Hardware Debugging’ then click on the ‘New’ button  (top left).



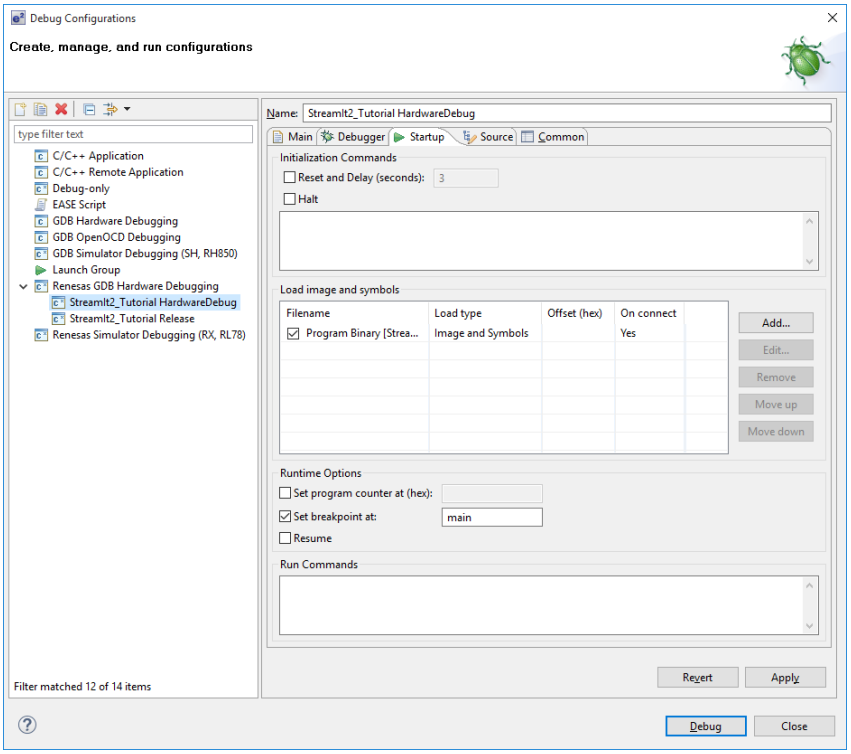
- The debug configurations control page will open. Observe the settings under each tab.
- Under the ‘Debugger’ tab, ensure the ‘Debugger hardware’ option is set to ‘J-Link ARM’.
- The ‘Target Device’ is preset to ‘R7S721031_DualSPI’.

  Note:

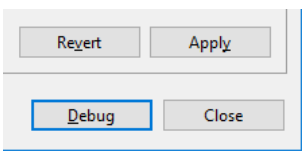
  Do **not** modify any settings.



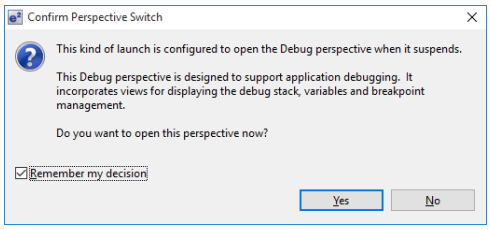
- A security dialog may appear indicating that the Windows Firewall has blocked some features of the eclipse platform.
- Under the text ‘Allow eclipse to communicate on these networks’, ensure the check box next to ‘Private networks, such as my home or work network’ is ticked. Click ‘Allow access’.
- Select the ‘Startup’ tab.
- Ensure the ‘Runtime Option’ ‘Set breakpoint at:’ is specified as ‘main’.



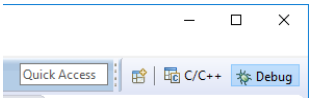
- Click on ‘Debug’.



- Before downloading the code a dialog box will appear asking if you would like to switch to the 'e^2 studio Debug perspective'. If you agree click 'Remember my decision' to prevent this dialog box from appearing in future, then click 'Yes'.



- e^2 studio will load the new perspective, which is optimised for debugging.
- To change back to the default 'C/C++' perspective, from the menu bar select Window -> Perspective -> Open Perspective ->Other
- The 'Open Perpsective' dialog box will appear. Click on the desired perspective to select it then 'OK'.
- Alternatively, click on the button within the top right corner of the screen, as shown opposite, and select the 'C/C++' perspective.



3. Reviewing the Tutorial Program

This section will look at each section of the tutorial code and basic debugging functionality in e^2 studio.

3.1 Main Functions

This section will look at the program code called from within the main() function, and how it works.

- Start a debug session for the tutorial program as described in the previous section. The debugger should connect and the program will be stopped on the first line of the main() function as shown in the screenshot opposite.
- Click on the line containing the 'flashled()' function call in 'main()' to position the cursor. Right-click and select 'Run to Line' to execute the program up to this line.

  The 'R_LCD_SetupScreen()' function call enables and configures the LCD, and the 'R_LCD_DisplayText()' function is used to write "Stream it! v2" on the top line and then "Tutorial Sample" below.

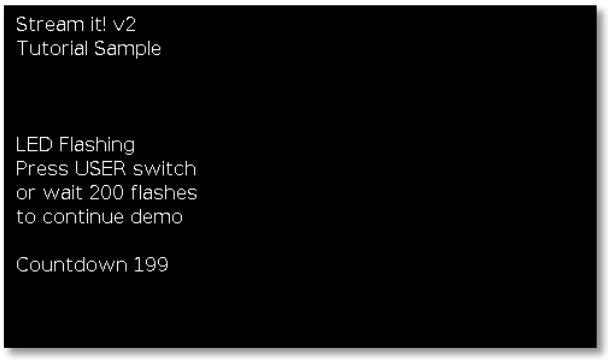
```
127          * Function Name: main
134          int8_t main(void)
135 20040598 {
136 200405a4     char strdata[32] = "";
137              uint16_t current_adc_percent;
138
139              /* Initialise the Colour LCD display */
140 200405d0     R_LCD_SetupScreen();
141
142              /* Initialise direct connected LED */
143 200405d4     R_LED_Init();
144
145              /* Initialise user switch */
146 200405d8     R_SWITCH_Init(true);
147
148 200405e0     R_LCD_DisplayText(0, (char *) " Stream it! v2");
149 200405f0     R_LCD_DisplayText(1, (char *) " Tutorial Sample");
150
151 20040600     flashled();
152 20040604     clear_display_area();
153
154 20040608     static_test();
155 2004060c     clear_display_area();
156
157 20040610     R_LCD_DisplayText(4, (char *) " Use P1 to set delay");
158 20040620     R_LCD_DisplayText(5, (char *) " using timer OSTM ");
159 20040630     R_LCD_DisplayText(6, (char *) " when flashing LEDs");
160
161              /* Initialise OS timer (channel 0) */
162 20040640     R_OSTM_Init(DEVDRV_CH_0, OSTM_MODE_INTERVAL, 500);
163
164              /* Start OS timer (channel 0) */
165 20040650     R_OSTM_Open(DEVDRV_CH_0);
```

- Set a breakpoint on the 'static_test()' function call by double-clicking in the blue breakpoint column.

```
151 20040600     flashled();
152 20040604     clear_display_area();
153
154 20040608     static_test();
155 2004060c     clear_display_area();
```

- Click the ‘Step Into’ button  to step into the ‘flashled()’ function.
- Click the ‘Resume’ button  to resume program execution.
- The program will now run the flashled() function. This function periodically polls the user switch and flashes the LED 200 times (as specified in the loopcount variable) or until the user switch has been pressed.

```
263             * Function Name: flashled
268             static void flashled(void)
269 20040970     {
270                 char strdata[256];
271 2004097c         uint16_t loopcount = 200;
272                 uint32_t delay;
273
274 20040984         R_LCD_DisplayText(5, (char *) " LED Flashing");
275 20040994         R_LCD_DisplayText(6, (char *) " Press USER switch");
276 200409a4         R_LCD_DisplayText(7, (char *) " or wait 200 flashes");
277 200409b4         R_LCD_DisplayText(8, (char *) " to continue demo");
278
279 200409c4         R_LED_Off();
280
281 200409c8         while (loopcount > 0)
282                 {
283 200409cc             sprintf(strdata, " Countdown %d ", --loopcount);
284
285                     /* Display the application name on the PMOD LCD */
286 200409f0             R_LCD_DisplayText(10, (char *) strdata);
287
288 20040a00             delay = 10000000u;
289 20040a0c             while (--delay)
290                     {
291 20040a10                 __asm__("nop");
292                     }
293
294                     /* Toggles the LEDs after a specific delay */
295 20040a2c             R_LED_Toggle();
296
297
298 20040a30             if (g_switch_press_flg)
299                     {
300 20040a48                 g_switch_press_flg = 0;
301 20040a58                 loopcount = 0;
302                     }
303                 }
304
305 20040a6c         R_LED_Off();
306 20040a70     }
308             * End of Function flashled
```



The LCD screen during flashing LED countdown

- The program counter should come to a halt at the static_test() function.
- Step into the function by clicking the ‘Step Into’ button  . Alternatively, press [F5].

```
151 20040600     flashled();
152 20040604     clear_display_area();
153
154 20040608     static_test();
155 2004060c     clear_display_area();
```

- Press [F7] or ‘Step Out’  to execute the static_test() function.
- Observe the string on the bottom line of the LCD change one character at a time from ‘STATIC’ to ‘TESTTEST’ as the ‘static_test()’ function is executed.
- After all characters have been changed, the LCD bottom line will return to displaying ‘STATIC’.

```
197          * Function Name: static_test
207          static void static_test (void)
208 20040794 {
209 200407a0     char strdata[32] = "";
210
211
212 200407cc     uint8_t     ui_count = 0;
213
214
215 200407d4     char        c_str[] = "STATIC  \0";
216
217
218 200407ec     const char  c_replace[] = "TESTTEST\0";
219
220 20040804     R_LCD_DisplayText(4, (char *) " Static Test");
221 20040814     R_LCD_DisplayText(5, (char *) " Initialise c_str");
222 20040824     R_LCD_DisplayText(6, (char *) " Replace contents ");
223 20040834     R_LCD_DisplayText(7, (char *) " of c_str with ");
224 20040844     R_LCD_DisplayText(8, (char *) " that of ucReplace");
225 20040854     R_LCD_DisplayText(9, (char *) " ucStr    ='STATIC  '");
226 20040864     R_LCD_DisplayText(10,(char *) " ucReplace='TESTTEST'");
227
228
229 20040874     sprintf(strdata, " c_str = '%s'  ", c_str);
230 2004088c     R_LCD_DisplayText(12, (char *) strdata);
231
232
233 2004089c     delay();
234
235              /* Begin for loop which writes one letter of ucReplace to the LCD at a time
236                 The nested while loops generate the delay between each letter change */
237 200408a0     for (ui_count = 0; ui_count < 8; ui_count++)
238          {
```

LCD screen during string character replacement

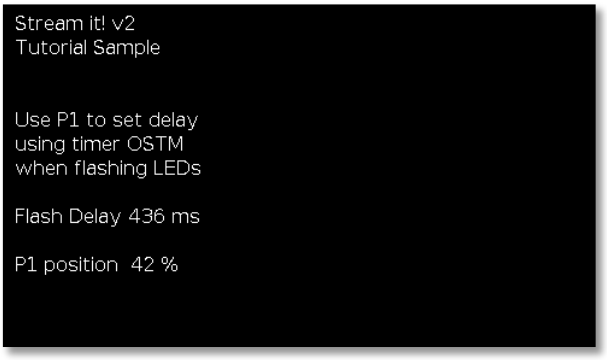
- The debugger will stop the program at the clear_display_area() function. Press F6 or click the 'Step Over' button  to execute this function and clear the display.

```
152 20040604     clear_display_area();
153
154 20040608     static_test();
155 2004060c     clear_display_area();
156
157 20040610     R_LCD_DisplayText(4, (char *) " Use P1 to set delay");
158 20040620     R_LCD_DisplayText(5, (char *) " using timer OSTM ");
159 20040630     R_LCD_DisplayText(6, (char *) " when flashing LEDs");
160
161              /* Initialise OS timer (channel 0) */
162 20040640     R_OSTM_Init(DEVDRV_CH_0, OSTM_MODE_INTERVAL, 500);
163
164              /* Start OS timer (channel 0) */
165 20040650     R_OSTM_Open(DEVDRV_CH_0);
166
167              /* Initialise Analogue input (Potentiometer) on board */
168 20040658     R_ADC_Open();
169
170              while (1)
171              {
172 2004065c         R_ADC_Read();
173
174 20040660         current_adc_percent = ((g_adc_result / 1023.0) * 100.0);
175
176              /* Set the minimum flash rate if the ADC value is less than the minimum */
177 200406c0         if (g_adc_result < MIN_FLASH_RATE)
178                  {
179 200406d4             g_adc_result = MIN_FLASH_RATE;
180                  }
181
182 200406e4         sprintf(strdata, " Flash Delay %d ms  ", (uint16_t) g_adc_result);
183 20040708         R_LCD_DisplayText(8, (char *) strdata);
184
185 20040718         sprintf(strdata, " P1 position %3d %%  ", (int16_t) current_adc_percent);
186 20040734         R_LCD_DisplayText(10, (char *) strdata);
187
188 20040744         OSTM0.OSTMnCMP = (uint32_t)(P0_CLOCK_FREQUENCY_kHz * (g_adc_result + 1));
189 20040790     }
190              return (0);
191          }
193          * End of function main
```

- Open src/renesas/peripherals/internal/ r_ostm_userdef.c and scroll to the bottom of the file to the sample_ostm0_interrupt() function.

```
215          * Function Name: sample_ostm0_interrupt
224          static void sample_ostm0_interrupt (uint32_t int_sense)
225 2004f79c {
226 2004f7ac     R_INTC_Disable(INTC_ID_OSTM0TINT);
227
228 2004f7b4     R_LED_Toggle();
229
230 2004f7b8     R_INTC_Enable(INTC_ID_OSTM0TINT);
231 2004f7c0 }
```

- Set a breakpoint on the first line of code inside the sample_ostm0_interrupt() interrupt handler.
- Continue to execute the program by clicking the 'Continue' button.
- The program will halt at the breakpoint due to the timer's period elapsing.
- Remove the breakpoint by double clicking on the breakpoint in the blue breakpoint column. Continue to execute the program by clicking the 'Continue' button.



LCD display while potentiometer P1 controls the LED flash rate

For further details regarding hardware configuration, please refer to the Stream it! - RZ User's Manual and the RZA1L Group Hardware Manual.

4. e² studio

The Quick Start Guide will have directed you to complete the installation of e² studio. In case of difficulty this section will give instructions to obtain e² studio.

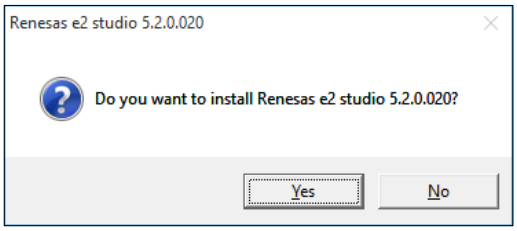
If e² studio is already installed, please follow section 4.2. However, for first time installation please follow the procedure described in section 4.1.

Section 4.1 gives instructions on installing e² studio version 5.2. It is recommended to use the latest version of e² studio as available on the web site.

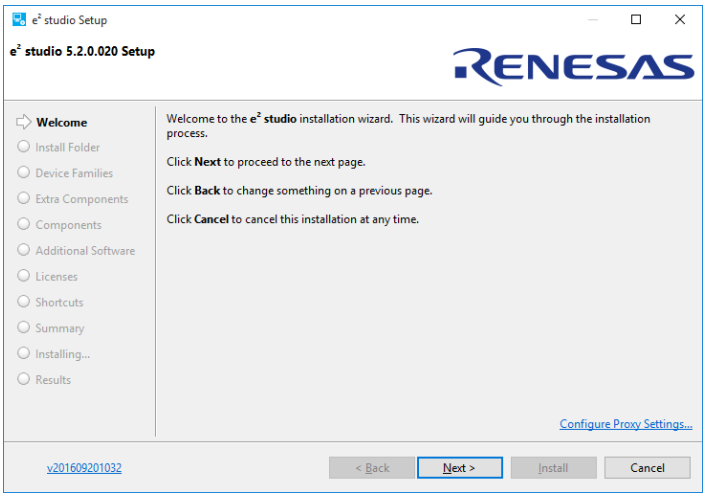
4.1 e² studio Installation

1. The latest e² studio installer can be acquired from the Renesas website at https://www.renesas.com/en-eu/products/software-tools/tools/ide/e2studio.html

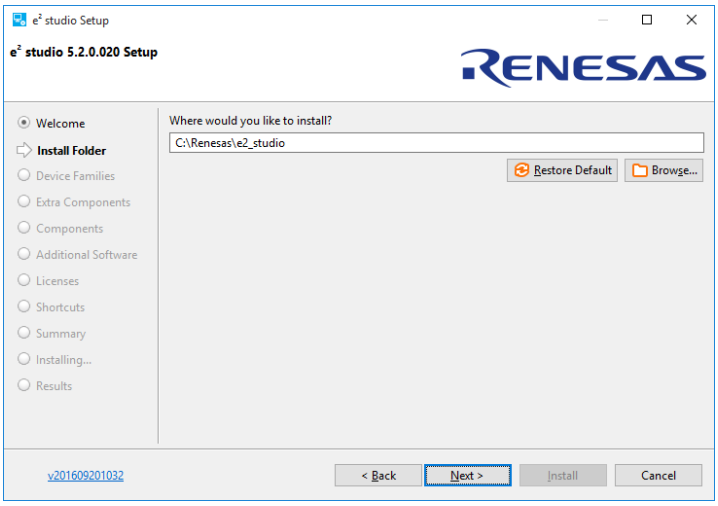
2. Once downloaded, double click on the application. A window will then pop-up, asking if you want to install Renesas e² studio (note the version number in the dialog will change). Click 'Yes'.



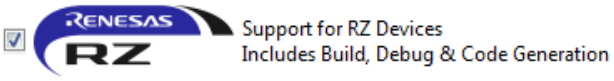
3. Once fully extracted, the e² studio installation wizard will guide you through the installation process. On the 'Welcome' tab click 'Next >'.



4. In the 'Install Folder' page, insert the path of a folder in which it is desired to be the root location for e² studio. It is suggested to keep the default path. To continue click 'Next >'.



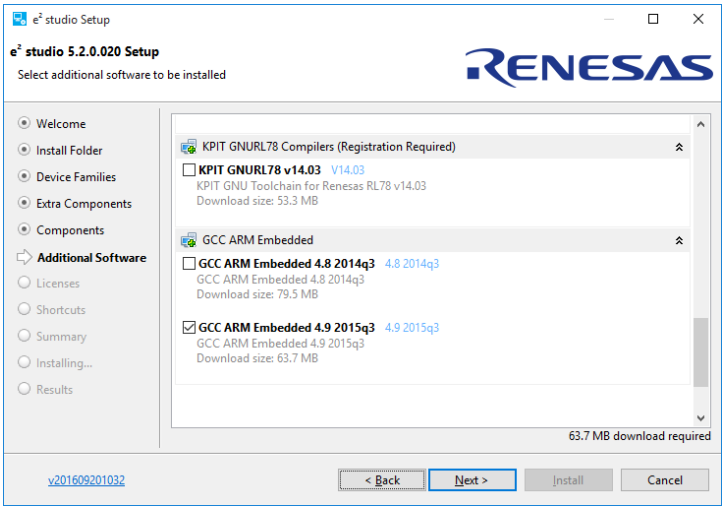
5. In the ‘Device Families’ page, ensure that the RZ family has been selected. It may also be desired to select support for other devices. Once selected, click ‘Next >’.



6. In the ‘Extra Components’ page you can select support needed for your development needs. To continue click ‘Next >’.

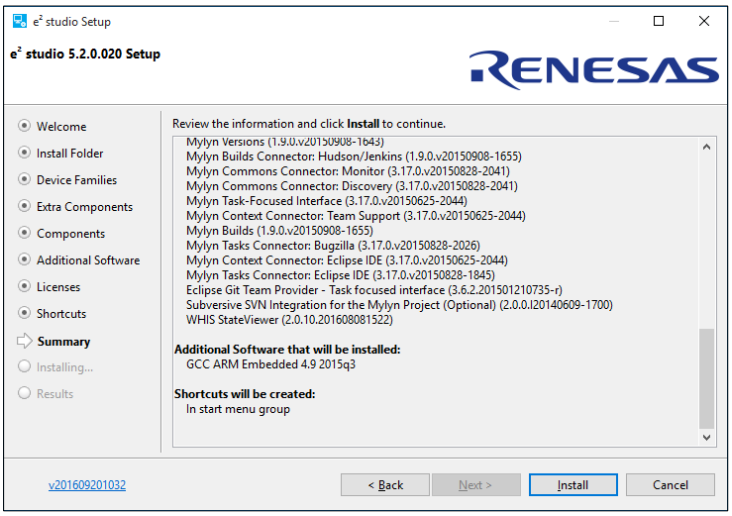
7. The ‘Components’ page will give the option to install optional components. It is recommended to ensure all are selected and to click ‘Next >’.

8. In the ‘Additional Software’ tab, ensure that ‘GCC ARM Embedded 4.9 2015q3’ is selected. Click ‘Next >’.



9. In the ‘Licenses’ page ensure to read and accept the Software Agreement to continue. Click ‘Install’.

10. The ‘Summary’ page will give an overview of the components of the installation. Click ‘Install’ to start the installation process.



11. Once the installation process has finished click ‘OK’.
To open e² studio please follow the instructions below.

- Start e² studio
  Windows™ 7: Start Menu > All Programs > Renesas Electronics e2studio > e2 studio

Windows™ 8 / 8.1: From Apps View, click 'e² studio' icon.
Windows™ 10: Start Menu > All apps > Renesas Electronics e2studio > e2 studio

- In the 'Select a workspace' dialog box, browse to a suitable location and enter a folder name to save your new workspace. Click 'OK' to continue.
- On the 'There are no new toolchains available for integration' message box, click 'OK'.
- In the e² studio 'Welcome' screen, click the 'Go to the workbench' arrow icon, on the far right.

- Code Generator Registration window will pop up to register code generator. Click 'OK'.
- Once registered, another pop-up window will ask you to restart e² studio. Click 'OK'. e² studio will restart.

4.2 e² studio Update

To update e² studio both RZ support and the GNU ARM Embedded v4.9.3 compiler are to be installed. This is recommended to be done on e² studio version 4.3 or later.

To install the 'RZ support' please follow the instructions below:

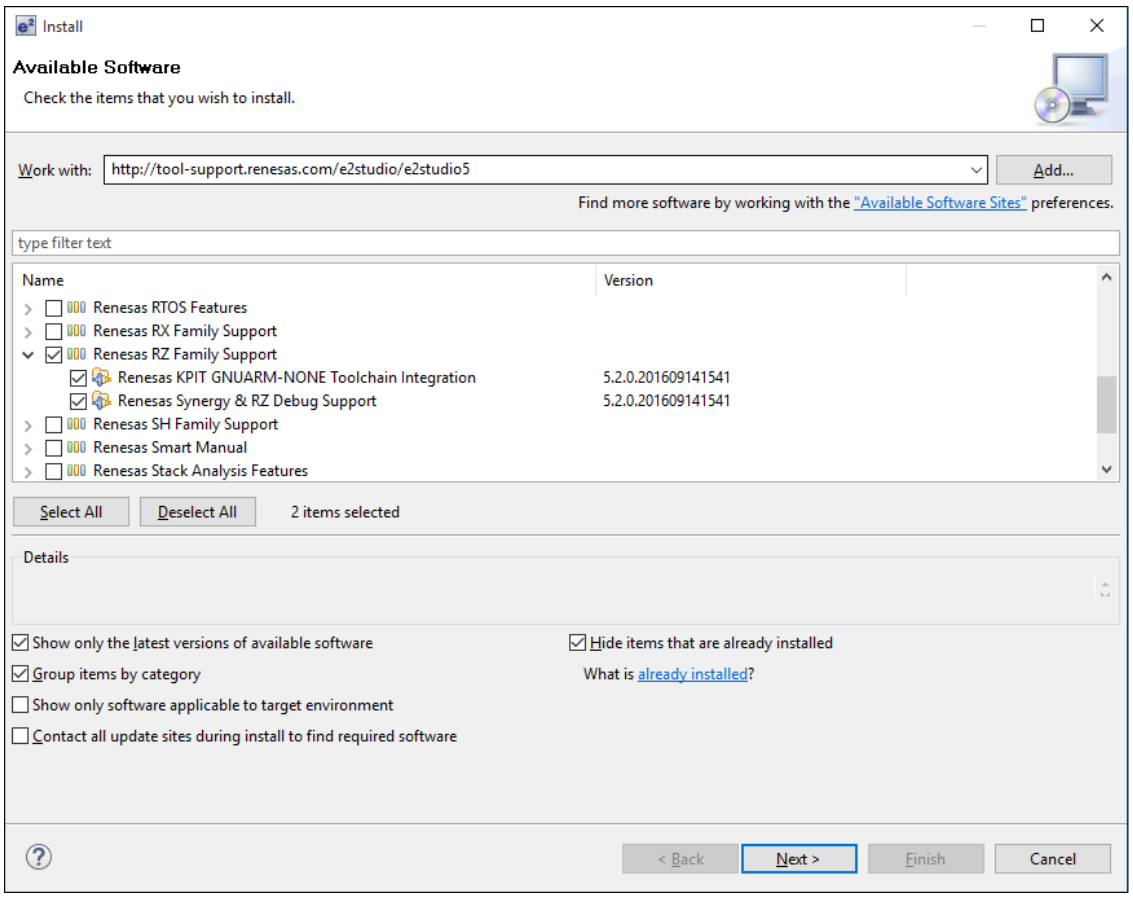
1. The RZ support can be installed through Renesas' tool support link. This can be achieved through

   Help -> Install New Software…

   Followed by inserting the following link in the 'Work with' box.

   http://tool-support.renesas.com/e2studio/e2studio5

2. Select the 'Renesas RZ Family Support' and click 'Next >'.

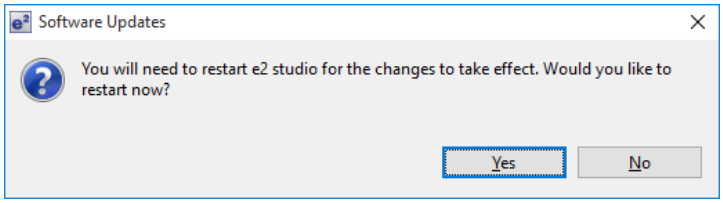


3. Installation details will then be shown. Click 'Next >'.

4. Read the ‘License text’ and select ‘I accept the terms of the license agreement’ to continue.

   A pop-up window will then ask you to restart e^2 studio. Click ‘Yes’.

5. Once restarted the installation process is complete.

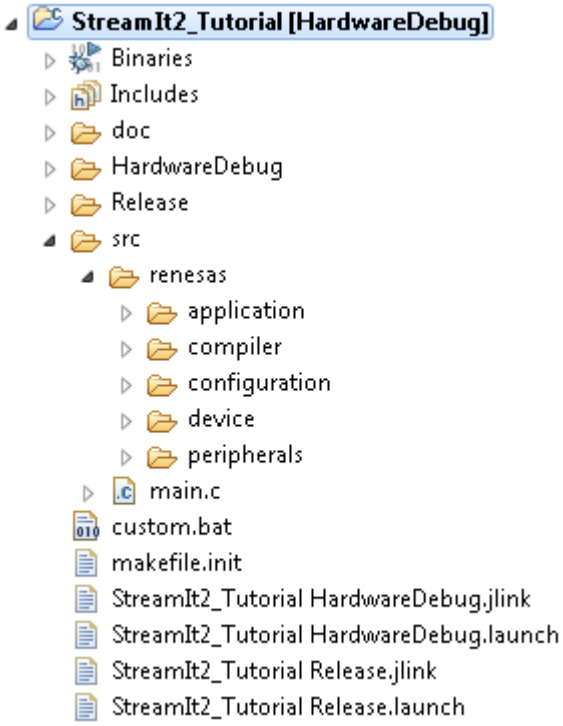


5. Project Details

This section details the sample project layout, components used and execution cycle.

5.1 Project Layout

The project layout as shown in e^2 studio is as below:



The following folders contain useful or user modifiable contents:

- doc Text file detailing simple download instructions and links to documentation
- HardwareDebug When built, stores the build files for the debug configuration
- Release When built, stores the build files for release configuration
- src Source code for project. All user modifiable code is located in this sub folder

Note the configuration folders also store a bin file which can be used in conjunction with the StreamIt2_QSPI_Loader.
Please refer to the relevant documentation on the RZ/A1LU QSPI Flash Boot Loader for details.

The layout of the src folder is as follows:

Name	Overview
application	Stores the demonstration application
compiler	Stores any files specific to the startup procedure of the microcontroller
peripherals	Stores the peripheral drivers required for this board. Internal folder stores the microcontroller peripheral drivers External folder stores the non-microcontroller peripheral drivers (e.g. LED, switch, etc.)
configuration	Configures the Stream it! - RZ version number
main.c	Contains the start of the user level application (main() function)

5.2 Runtime Environment

The following resources are used in the application:

Resource	Device	Function/Description
OSTM	OS Timer	Used for the delay when flashing the LED
VDC5	Video Display Controller 5	Driving the LCD display
ADC	Analogue to Digital converter	Digitising the position of the potentiometer

6. Additional Information

Technical Support

For details on how to use e^2 studio, refer to the help file by opening e^2 studio, then selecting Help > Help Contents from the menu bar.

For information about the RZA1L series microcontrollers refer to the RZA1L Group Hardware Manual.

Technical Contact Details

***Please refer to the contact details listed in section 5 of the Stream it! - RZ* "Quick Start Guide" (r12qs0013eg0100-rza1lu.pdf).**

Renesas Electronics Website:
https://www.renesas.com/

Inquiries:
https://www.renesas.com/contact/

This product's homepage, where additional documentation and source code can be found, is located at:
https://www.renesas.com/en-eu/solutions/key-technology/human-interface/rz-stream-it.html

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	20 Mar 2017	All	Original release
1.10	16 Nov 2020	All	Application Note Software Update.

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.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

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

(Rev.4.0-1 November 2017)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

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

Contact information

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