

RZ/A2M Group

RZ/A2M Software Package Quick Start Guide

1. Introduction

This is the Quick Start Guide for the RZ/A2M Software Package which works on RZ/A2M CPU board + RZ/A2M SUB board and the operation of Renesas e² studio.

This document describes how to run each executable sample project included in the package.

2. Preparation

2.1 Tool

RZ/A2M Software Package can be used on the following environment. Please check your environment before continuing.

Tools:

- IDE: e² studio 2021-04 Windows 64-bit product version or later
Available from
<https://www.renesas.com/software-tool/e-studio>
- Tool Chain: GNU ARM Embedded Toolchain 6-2017-q2-update
This tool is bundled in the IDE above. Available from
<https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/gnu-rm/downloads/6-2017-q2-update>
Refer to the following document for the details of installing e2 studio.
- [e2 studio Integrated Development Environment User's Manual: Getting Started](#)

Target Board:

RZ/A2M CPU board (RTK7921053C00000BE)

RZ/A2M SUB board (RTK79210XXB00000BE)

Bootloader:

This package includes the binary format bootloader program. If user would like to get the source code, download it from following website.

<https://www.renesas.com/rza2m-evaluation-board-kit>

2.2 Virtual Serial Port Connection

Connect CN5 on the RZ/A2M SUB board to a Windows™ PC, this provides a USB virtual serial port.

When the RZ/A2M SUB board is first connected, the PC will look for a suitable driver. This driver is installed during the installation process and the PC should automatically find and install it. The PC will report it is installing a driver and report a driver has been installed successfully. The COMx port number allocated to the virtual serial port can be confirmed in Windows™ Device Manager.

2.3 Serial Terminal

- Start a serial terminal program (such as PuTTY, HyperTerminal or Tera Term) using the following configuration:

Baud Rate: 115200

Data Bits: 8

Parity: None

Stop Bits: 1

Flow Control: None

COM Port: As shown in Windows™ Device Manager.

3. Trying sample application

3.1 Importing Software Package into IDE

This package is distributed as an archive file. Build project of this package can be imported into e² studio from the Project Import Menu. User can import the project to e² studio by the following procedure in this section.

- Obtain the package to use.
- Extract the contents of the package.
- Extract the individual projects to a short path.
- Launch e² studio from the start menu.
- Set the top directory which has each sample project sub-directory for the workspace directory. These 2 steps are shown in Figure 3-1.

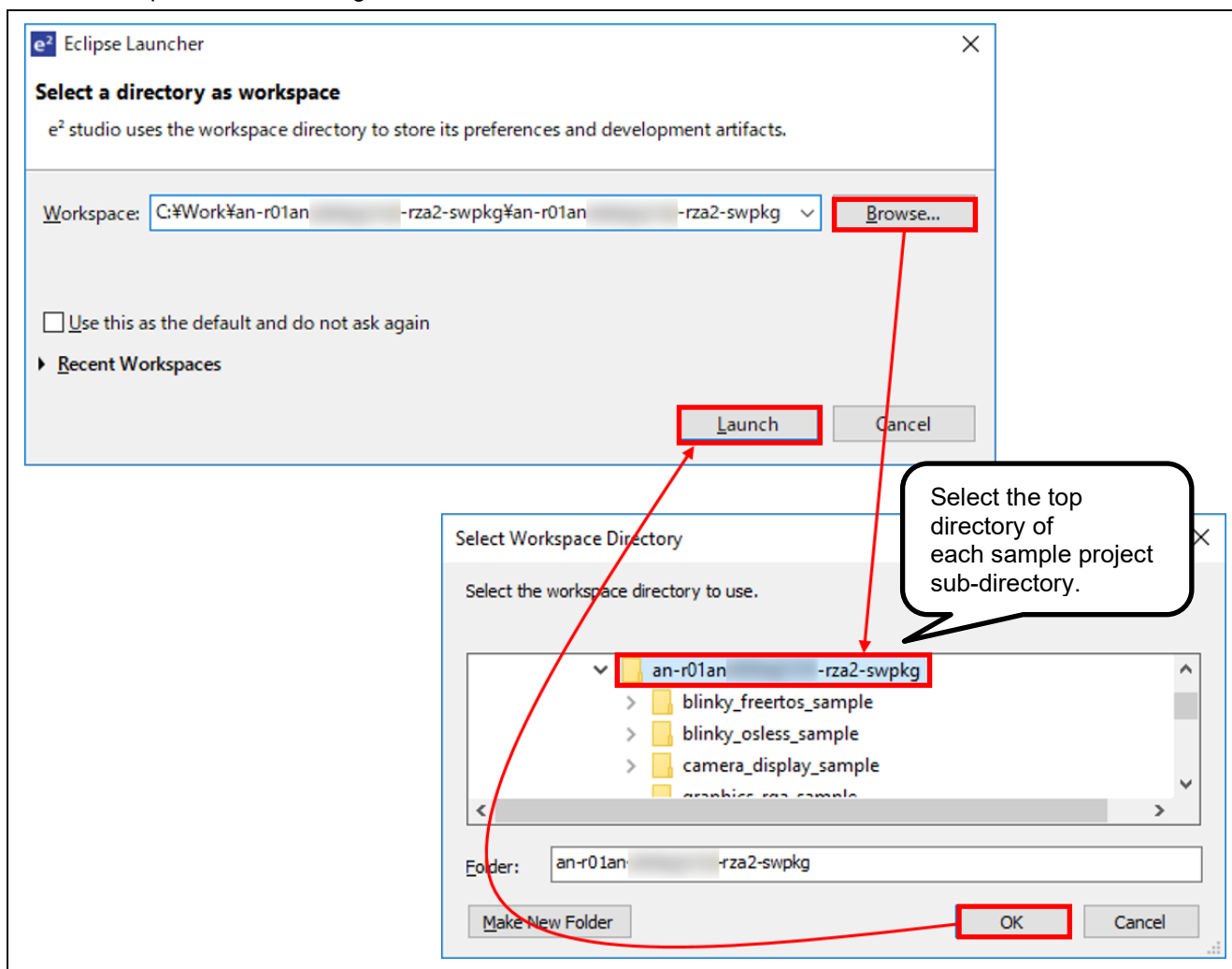


Figure 3-1 e2 studio launching

— In the e² studio welcome screen, click 'Workbench'. This is shown in Figure 3-2.

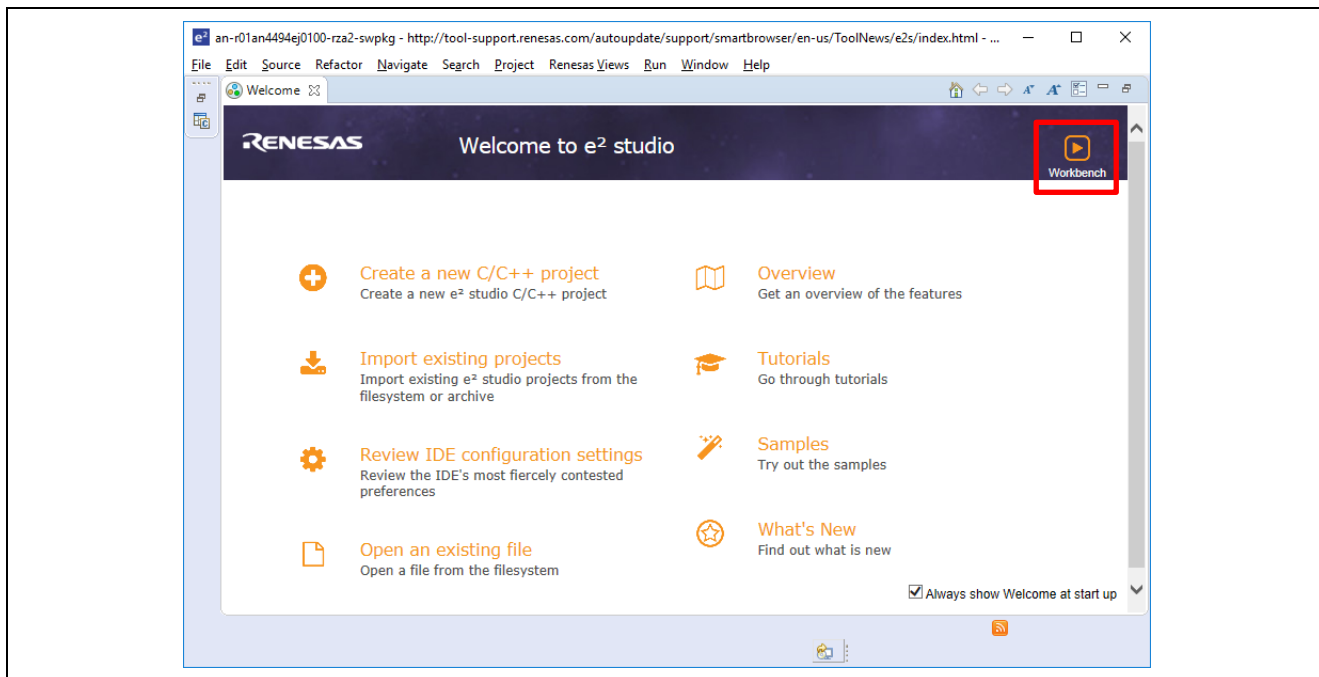


Figure 3-2 Position of 'Workbench' switch

— Right-click in the Project Explorer window and select 'Import'. This is shown in Figure 3-3.

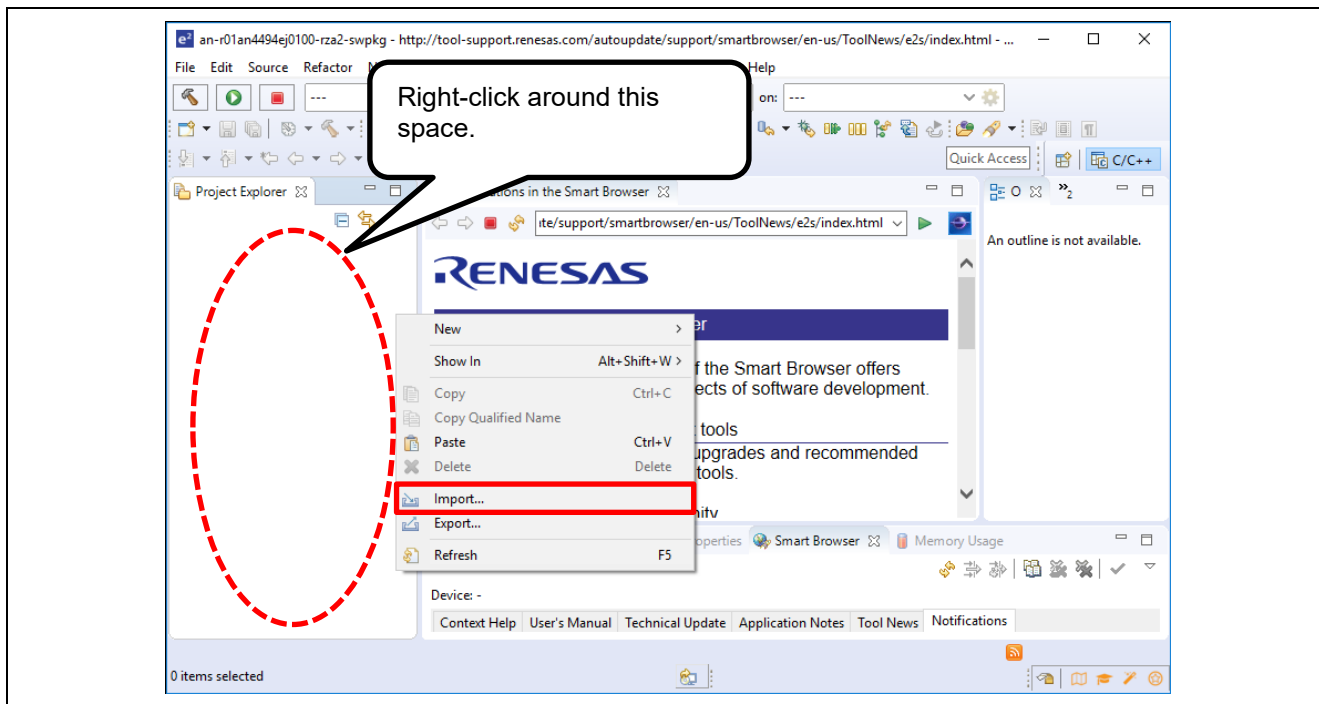


Figure 3-3 Selecting 'Import'

- Under 'Import' window, select General > Existing Projects into Workspace and click 'Next'. This is shown in Figure 3-4.

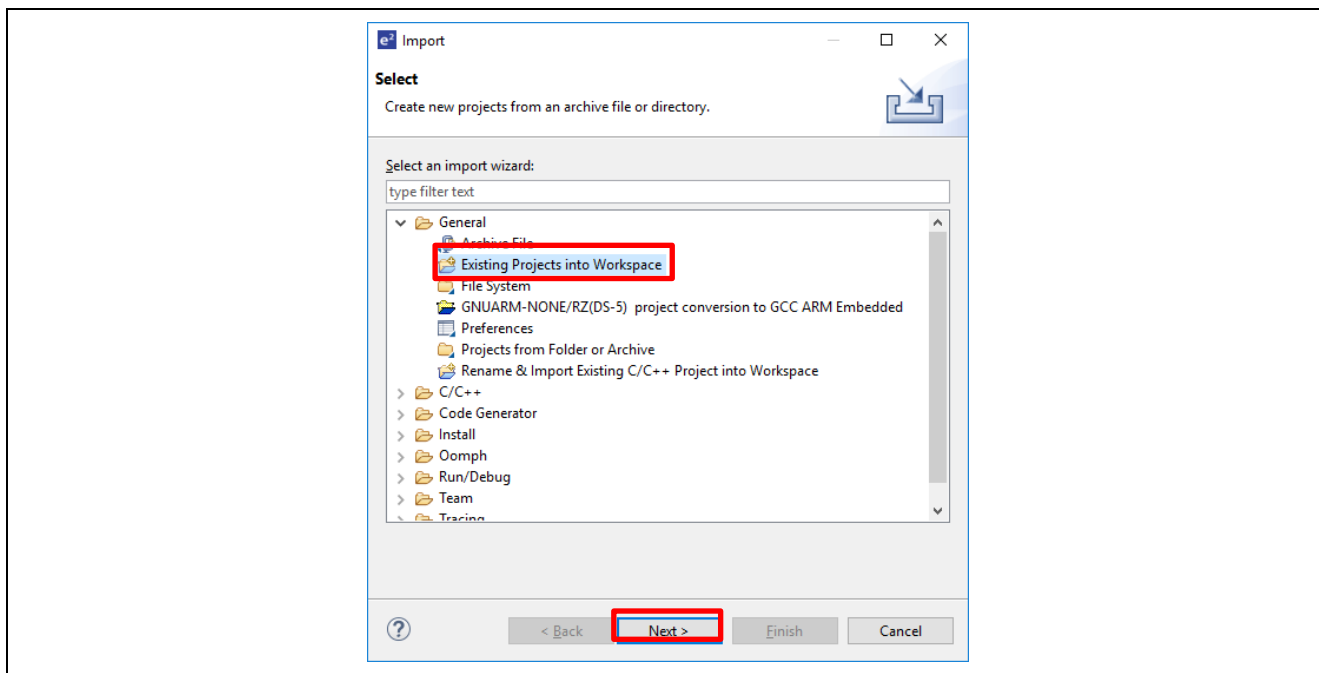


Figure 3-4 Menu of 'Import' window

- Select "Browse" at the right of "Select root directory:", and "Browse for Folder" dialog box will be appeared.
- Click "OK". These 2 steps are shown in Figure 3-5.

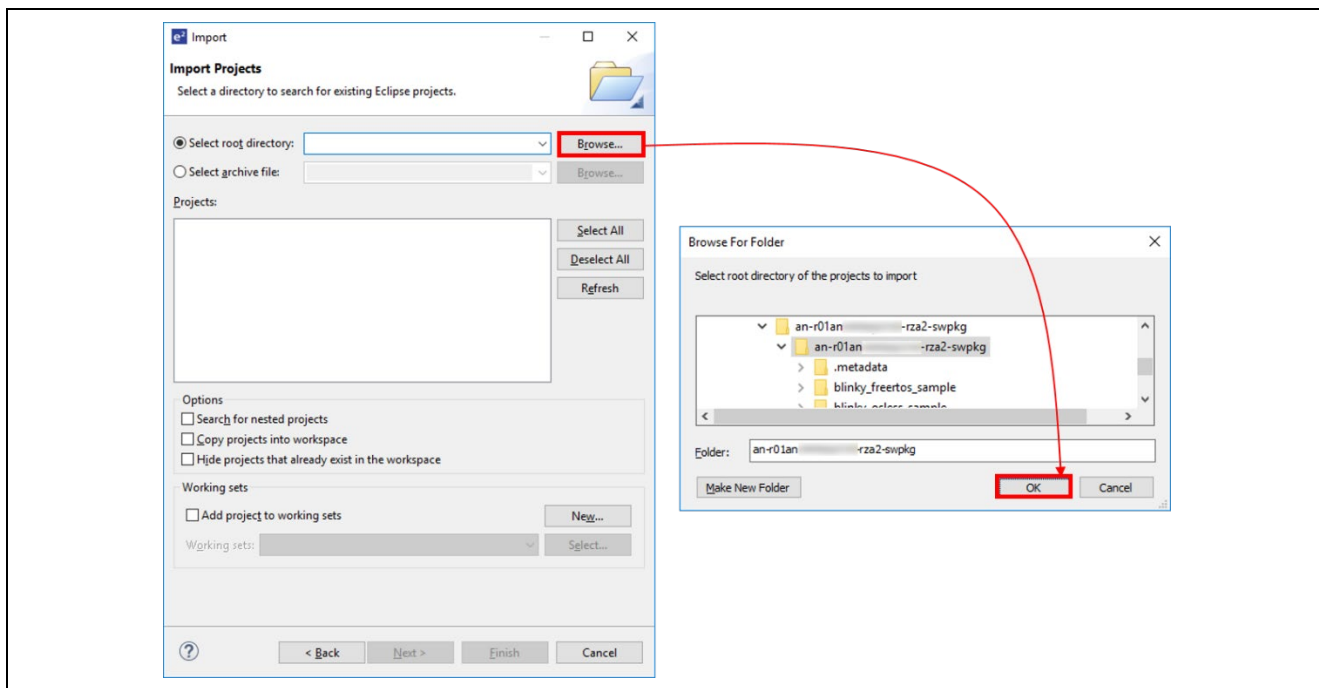


Figure 3-5 Select root directory

- Confirm your target project is checked, then click 'Finish'. This is shown in Figure 3-6.
(Note: Projects in Figure 3-6 are just sample. From here, please read the project name as your target project name.)

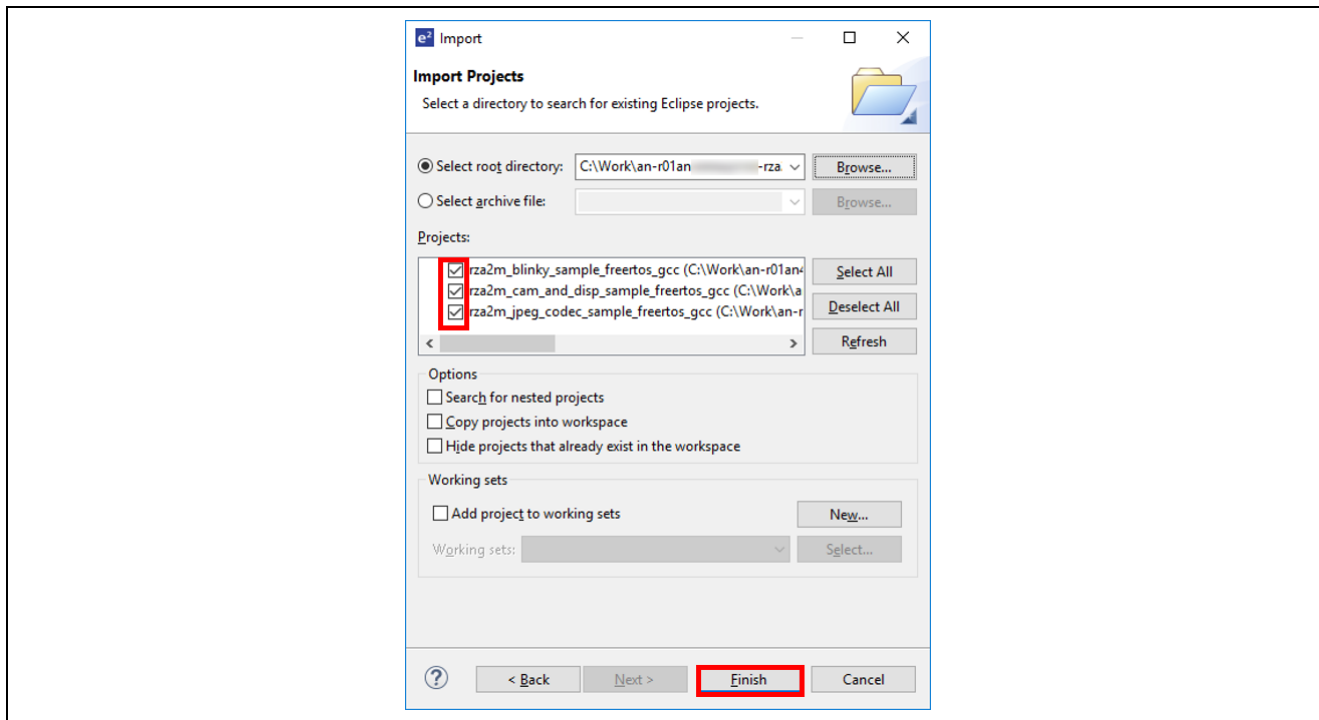


Figure 3-6 Import target project

- Now, target projects are imported, and user can see them in the Project Explorer window. This is shown in Figure 3-7.

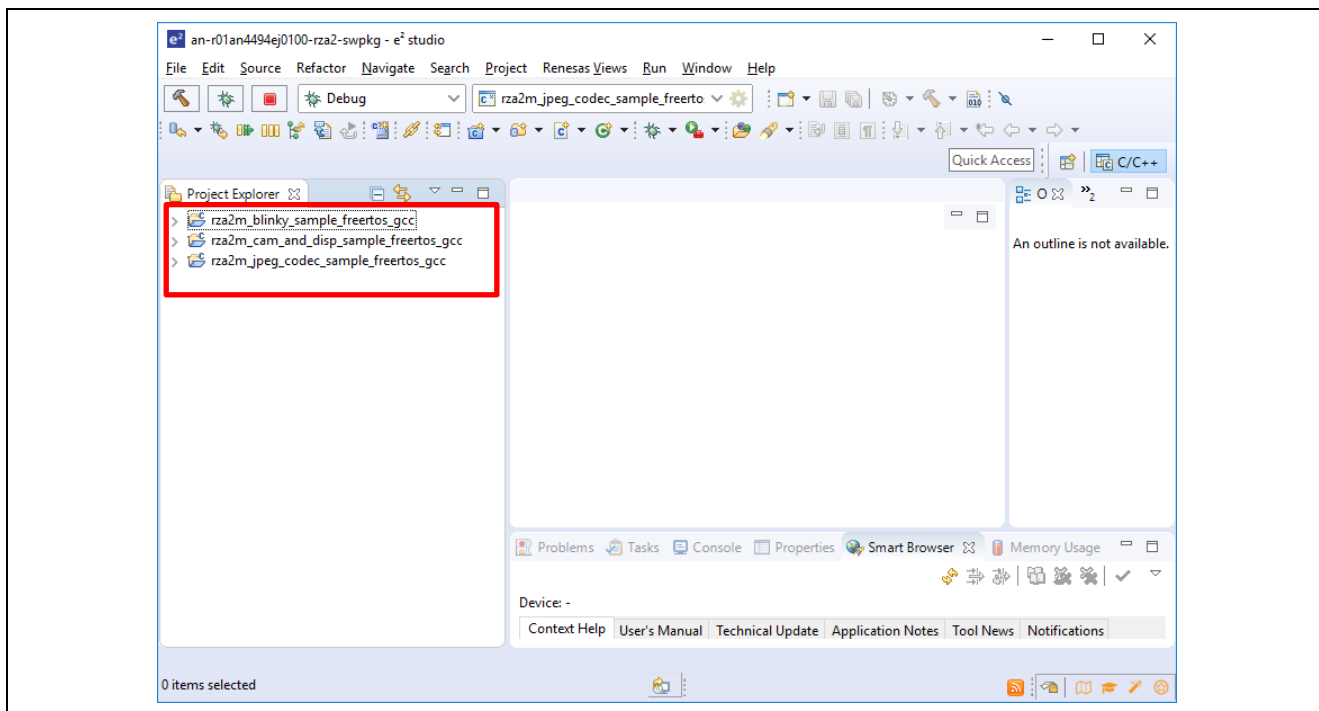


Figure 3-7 Confirmation on 'Project Explorer' window

3.2 Build and Download to target board

- Select your target project by left clicking on it, then click the arrow next to build button (hammer icon) and select 'HardwareDebug' from the drop-down menu. From next time, user can build the project by this Build button (hammer icon). This is shown in Figure 3-8.

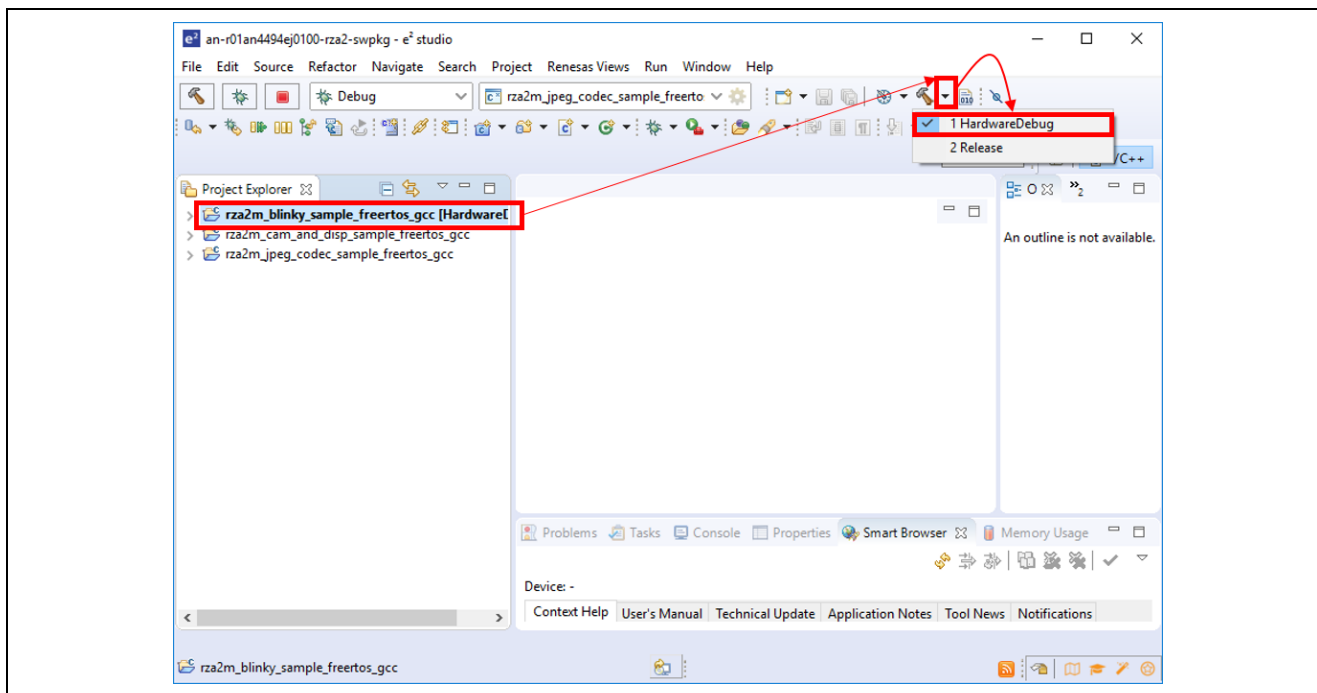


Figure 3-8 Build the target project

- e2 studio tool build the project, and the build status can be confirmed in Console window (Note: Please mind the length of your workspace path. If the path is too long, there is a possibility of build error.) This is shown in Figure 3-9.

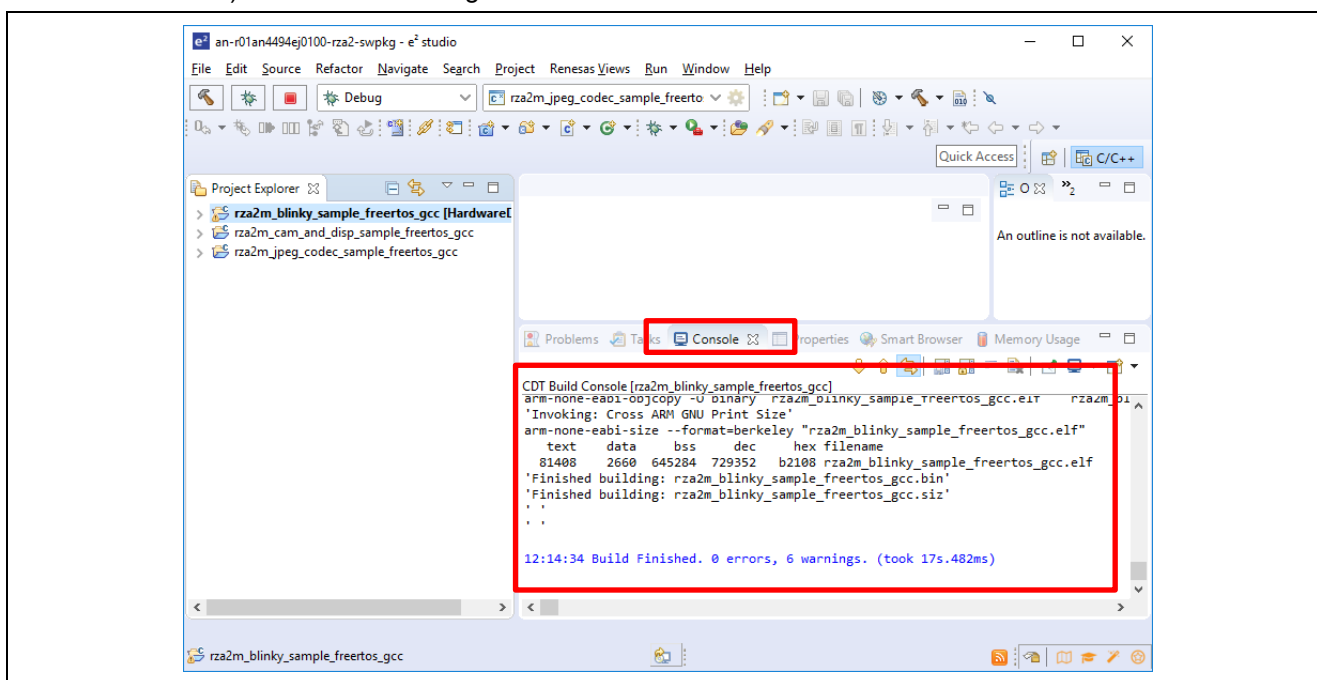


Figure 3-9 Confirmation build status

- After the build is completed, the Debug Configuration dialog can be opened by clicking "arrow" next to the debug button (insect icon) and selecting "Debug Configuration" from the drop-down menu. Debugging starts when user selects each target project in "Renesas GDB Hardware Debugging" and click "Debug" button. From next time, user can start debugging with just the click of the debug button (insect icon).

(Note: Ensure the target project is selected in Project Explorer, and the "Debug" button will not be enabled if the build is not completed normally.) This is shown in Figure 3-10.

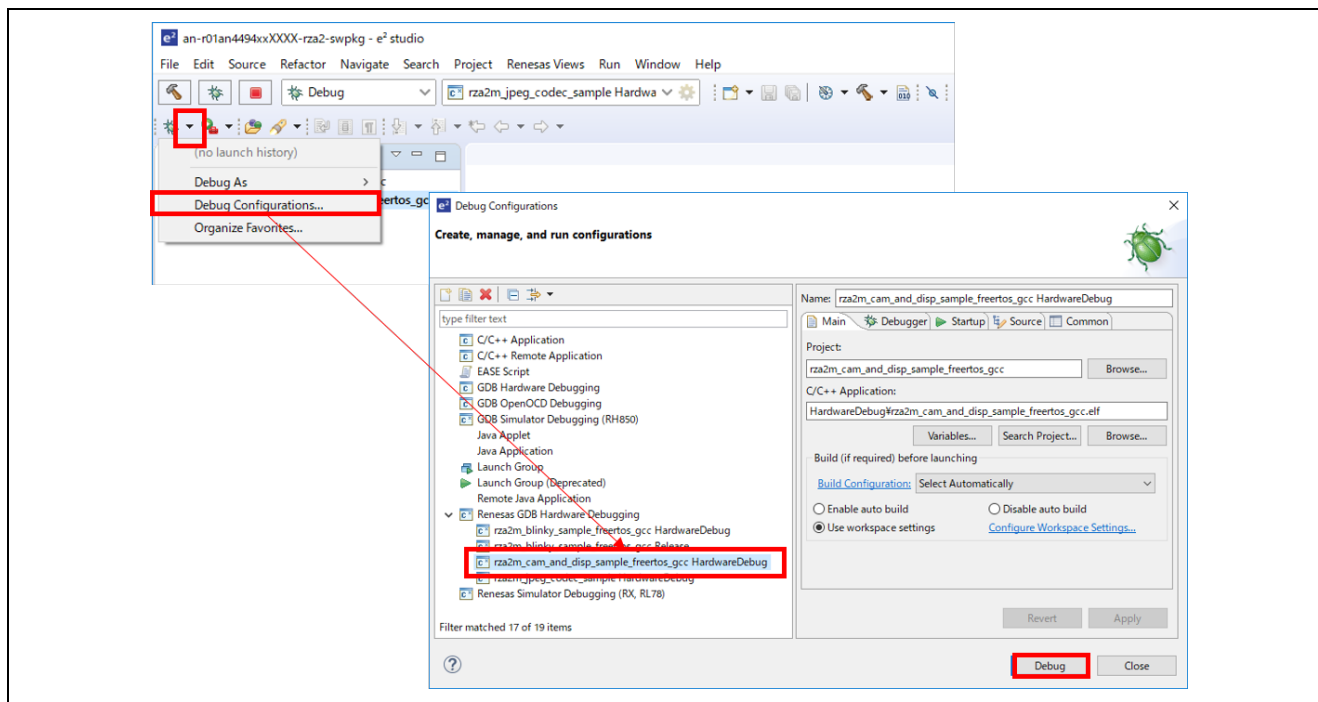


Figure 3-10 Start to debug

- The target program will be downloaded and e² studio may ask user to 'Confirm Perspective Switch', click 'Yes'. This is shown in Figure 3-11.

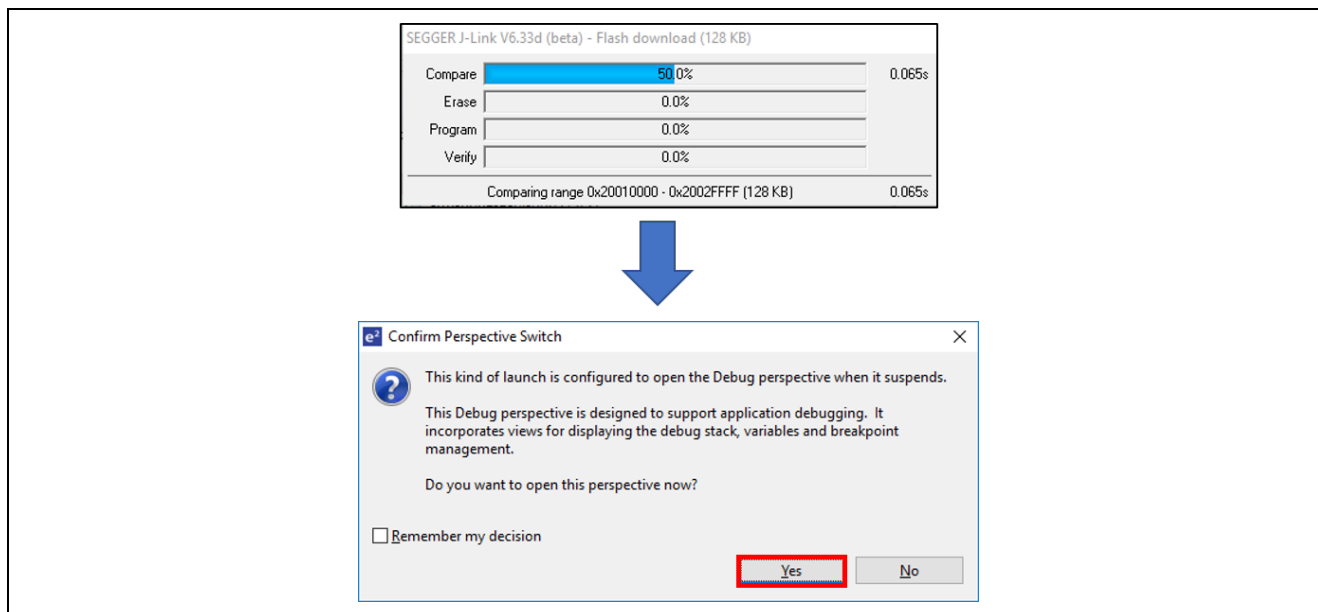


Figure 3-11 Downloading the target program

- Once the target program has been downloaded, click the 'Resume' button to run the target program. This is shown in Figure 3-12.

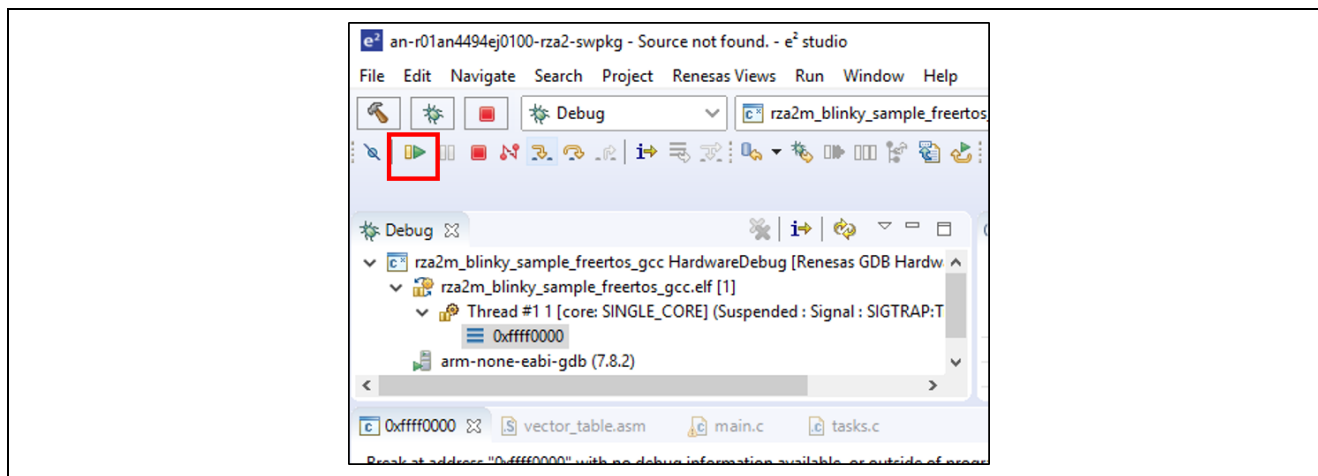


Figure 3-12 Run the target program

If the target project's debug configuration has break setting in main like Figure 3-13, operation will break at the top of main function.

If this is the case in your target project, click the 'Resume' button to resume running the code. This is shown in Figure 3-13.

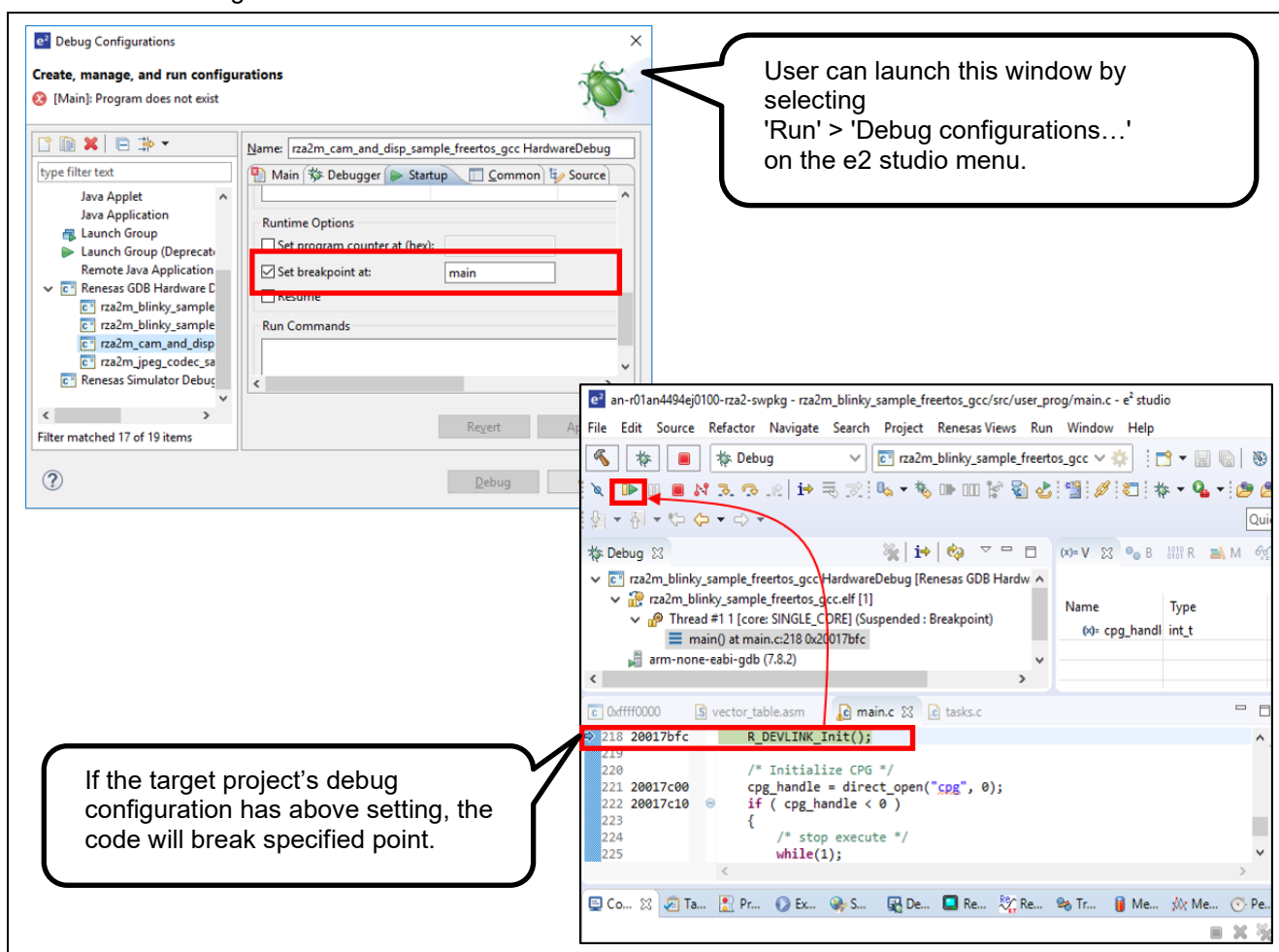


Figure 3-13 Behavior after running the target program

4. Adding drivers and middleware

This section describes how to integrate drivers, middleware into the project which included in this package. In RZ/A2M Software Package, the drivers and middleware are managed as components, and the components can be added by e2 studio.

Refer the sample projects bundled in RZ/A2M Simple Applications Package(R01AN4494) for examples of usage of each driver and each middleware.

Refer [RZ/A2M Smart Configurator User's Guide: e² studio](#) (R20AN0583) for the usage of Smart Configurator. e.g.) how to install drivers and middleware to e2 studio.

4.1 Components and related sample projects

Following table shows the components used in each sample project:

Package	Component (Explanation)		Sample Program																													
			silx(Silx Wi-Fi Libraries)	fatfs(FatFS filesystem)	r_vdc(VDC6 driver)	r_usbf_cdc(USB CDC driver)	r_usbf_basic(USB basic driver)	r_usb1_msc(USB ch1 MSC driver)	r_usb1_basic(USB ch1 basic driver)	r_usb0_msc(USB ch0 MSC driver)	r_usb0_basic(USB ch0 basic driver)	r_usbhid_basic(USBH ch0 HID driver)	r_ssif(SSIF driver)	r_sdhi_simplified(SDHI driver)	r_scifa(SCiFa driver)	r_nvapi(Video utility)	r_rtc(RTC driver)	r_rspi(RSPI driver)	r_riic(RIIC driver)	r_rga(Graphics Library "RGA")	r_ostm(OS timer driver)	r_mipi(MIPI driver)	r_lpm(LPM driver)	r_jcu(JCU driver)	r_hyperbus(Hyperbus driver)	r_gpt(GPT driver)	r_ether(Ethernet driver)	r_drp(DRP driver)	r_dmac(DMAC driver)	r_ceu(CEU driver)	r_cbuffer(Ring buffer)	r_adc(ADC driver)
simple applications	adc ^{*3}	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	blinky_freertos ^{*1}	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	blinky_osless ^{*2}	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	cam_and_disp ^{*3}	-	✓	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	dmac_scifa ^{*3}	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-
	drp_basic ^{*3}	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-
	drp_dynamic1 ^{*3}	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-
	drp_dynamic2 ^{*3}	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-
	drp_dynamic3 ^{*3}	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-
	drp_parallel ^{*3}	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-
	drp_simple_isp_sample1 ^{*3}	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-
	eeeprom_riic ^{*3}	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	ethernet ^{*3}	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-
	fw_update_boot ^{*3}	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-
	fw_update_sample ^{*3}	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	gpt-pwm ^{*4}	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-
	lpm ^{*4}	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-
	jpeg_codec ^{*3}	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	rtc ^{*4}	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-
	sdhi_fat ^{*3} ^{*4}	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	sprite_engine ^{*3}	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	ssif ^{*3}	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	touch_panel ^{*3}	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	usbh_msc_fat ^{*3} ^{*4}	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	usbh_hid ^{*3}	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	usbf_cdc ^{*3} ^{*4}	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	wifi_pmod_esp32 ^{*3}	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
2d_barcode	2d_barcode ^{*3}	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
iris	iris ^{*3}	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Graphics RGA	graphics ^{*3}	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Wi-Fi	wifi_sx_sdmac_eval ^{*3}	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	wifi_sx_sdmac ^{*3}	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Note ✓ : Component is used.

- : Component is not included.

*1 : "rza2m_blinky_sample_freertos_gcc" is the name displayed in e² studio.

*2 : "rza2m_blinky_sample_osless_gcc" is the name displayed in e² studio.

*3 : "rza2m_[SampleProgram]_sample_freertos_gcc" is the name displayed in e² studio.

*4 : "rza2m_[SampleProgram]_sample_osless_gcc" is the name displayed in e² studio.

4.2 Importing Software Package into IDE

- Open the project tree of target project in the Project Explorer window of e2 studio, and double click .scfg file in the project.
- Select 'Components' tab and add the target component from 'Add component' button.
- After adding the target component, click 'Generate Code' button.

The steps above are shown in Figure 4-1.

With this, the component is added to the target project's folder, "(Project Folder)\generate\sc_drivers" and "(Project Folder)\.settings\smartconfigurator". After adding the component, re-build the target project according to section 3.2.

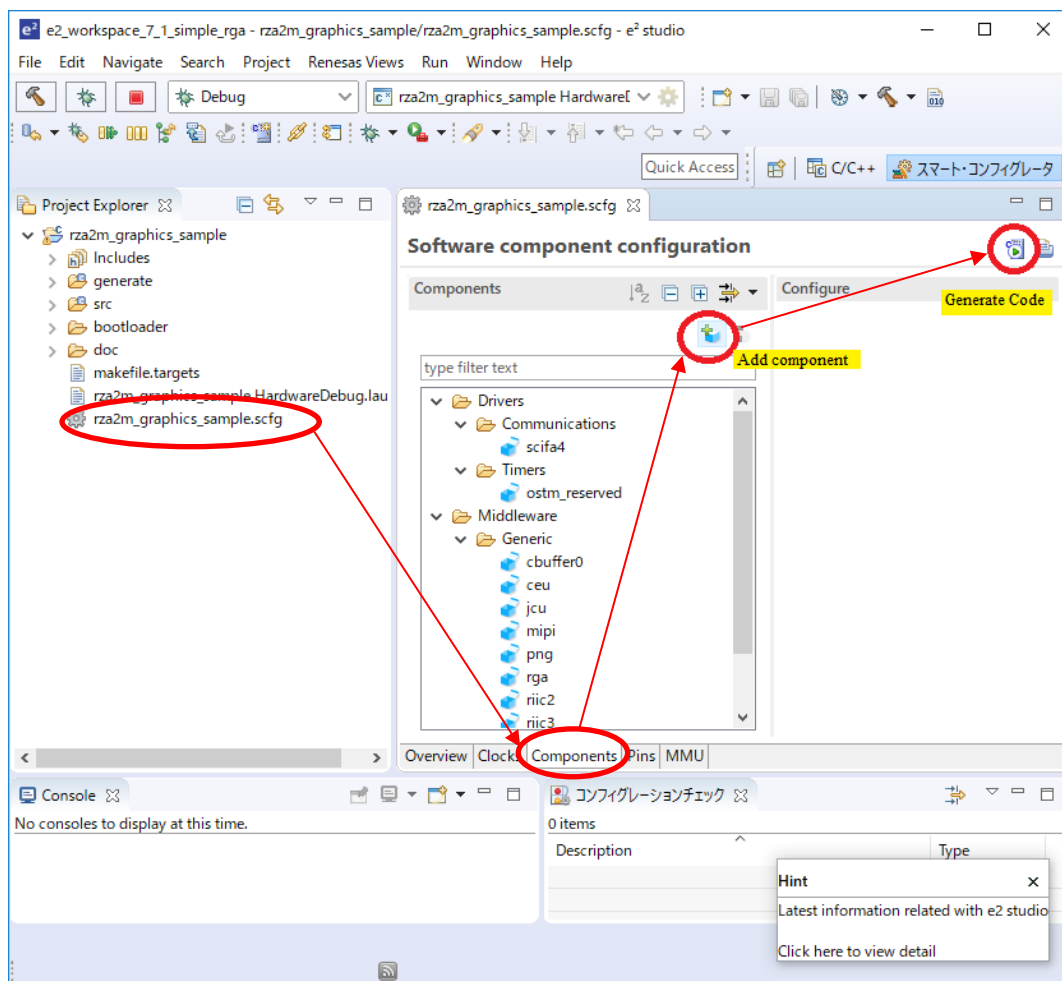


Figure 4-1 How to add components

4.3 How to integrate the new component

New component is integrated to a project by the following step

- Add component by the procedure shown in section 4.2.
- Configure the component by Smart Configurator.
- Generate the source code of the component.
- Confirm API functions or the name of header file declaring API functions from the document of the component.
- Use the API function name or the header file name to find where the component is used.
- Implement the source code into a project by referring the source file using the component.

As an example, it is shown that EEPROM reading/writing using RIIC3 implementation into "rza2m_blinky_sample_freertos_gcc".

- Open “rza2m_blinky_sample_freertos_gcc.scfg” file in “rza2m_blinky_sample_freertos_gcc” project. And add RIIC3 by the procedure shown in section 4.2.
- In the case using EEPROM implemented on the RZ/A2M Evaluation Board Kit, configure riic3 component as follows:
 - Change “Clock Frequency” to “400KHz”.
 - Change “RIIC3” to “Used”.
 - Change “RIIC3SCL Pin” to “Used”.
 - Change “RIIC3SDA Pin” to “Used”.

These settings are shown in Figure 4-2.

Property	Value
▼ Configurations	
# RIIC Transfer Mode	Master Mode
# Own Slave Address0 Enable	<input checked="" type="checkbox"/> Used
# Own Slave Address1 Enable	<input type="checkbox"/> Unused
# Own Slave Address2 Enable	<input type="checkbox"/> Unused
# Own Slave Address0	1
# Own Slave Address1	1
# Own Slave Address2	1
# Own Slave Address0 Length	7bit Address Mode
# Own Slave Address1 Length	7bit Address Mode
# Own Slave Address2 Length	7bit Address Mode
# SCL Clock Frequency	400KHz
# SCL Duty Cycle	50%
# SCL Rise Time (nsec)	0
# SCL Fall Time (nsec)	0
# Noise Filter	Unused
# SCL Timeout Mode	Unused
# RIIC Bus Format	I2C Format
# SMBus Host address detection	<input type="checkbox"/> Unused
# TEND Interrupt priority level (0 - 31)	9
# RDRF Interrupt priority level (0 - 31)	9
# TDRE Interrupt priority level (0 - 31)	9
# STOP Interrupt priority level (0 - 31)	9
# START Interrupt priority level (0 - 31)	9
# NACKF Interrupt priority level (0 - 31)	9
# AL Interrupt priority level (0 - 31)	9
# TMOF Interrupt priority level (0 - 31)	9
▼ Resources	
▼ RIIC	
▼ RIIC3	<input checked="" type="checkbox"/>
RIIC3SCL Pin	<input checked="" type="checkbox"/> Used
RIIC3SDA Pin	<input checked="" type="checkbox"/> Used

Figure 4-2 Configuration of riic3 component

— Press Generate “button”, then code will be generated. This is shown in Figure 4-3.

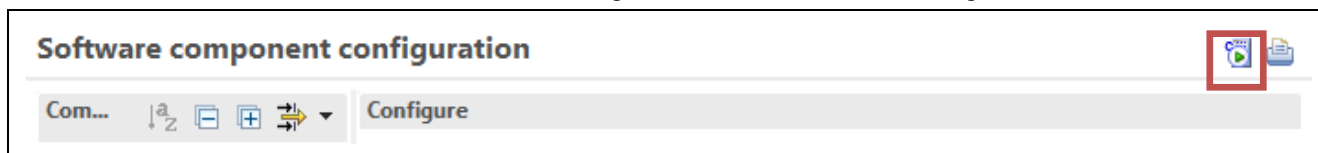


Figure 4-3 Code generation

— Generated files are listed in “Console” view. We can know the document of riic driver is generated in “generate\sc_drivers\r_riic\doc” folder by the list. This is shown in Figure 4-4.

```

M00000001: Code generation is started
M03000002: File generated:generate\drivers\r_cpg\inc\r_cpg_drv_sc_cfg.h
M04020001: File generated:generate\sc_drivers\r_riic\doc\r01an4645eg0100-rza2-riic-driver.pdf
M04020001: File generated:generate\sc_drivers\r_riic\inc\r_riic_drv_api.h
M04020001: File generated:generate\sc_drivers\r_riic\inc\r_riic_drv_link.h
M04020001: File generated:generate\sc_drivers\r_riic\inc\r_riic_drv_sc_cfg.h
M04020001: File generated:generate\sc_drivers\r_riic\inc\r_riic_hld_api.h
M04020001: File generated:generate\sc_drivers\r_riic\inc\r_riic_hld_prv.h

```

Figure 4-4 List of generated files

— In the document, it is described that API functions of riic driver are declared in r_riic_drv_api.h.

— By the table in section 4.1, it is described that RIIC driver is used in “rza2m_eeprom_riic_sample_freertos_gcc” project. Import “rza2m_eeprom_riic_sample_freertos_gcc” project to e² studio.

— Search the file in which “r_riic_drv_api.h” is included, in rza2m_eeprom_riic_sample_freertos_gcc project.

Or find the file in which API function is used.

Press “Search” menu and select “File...”. This is shown in Figure 4-5.

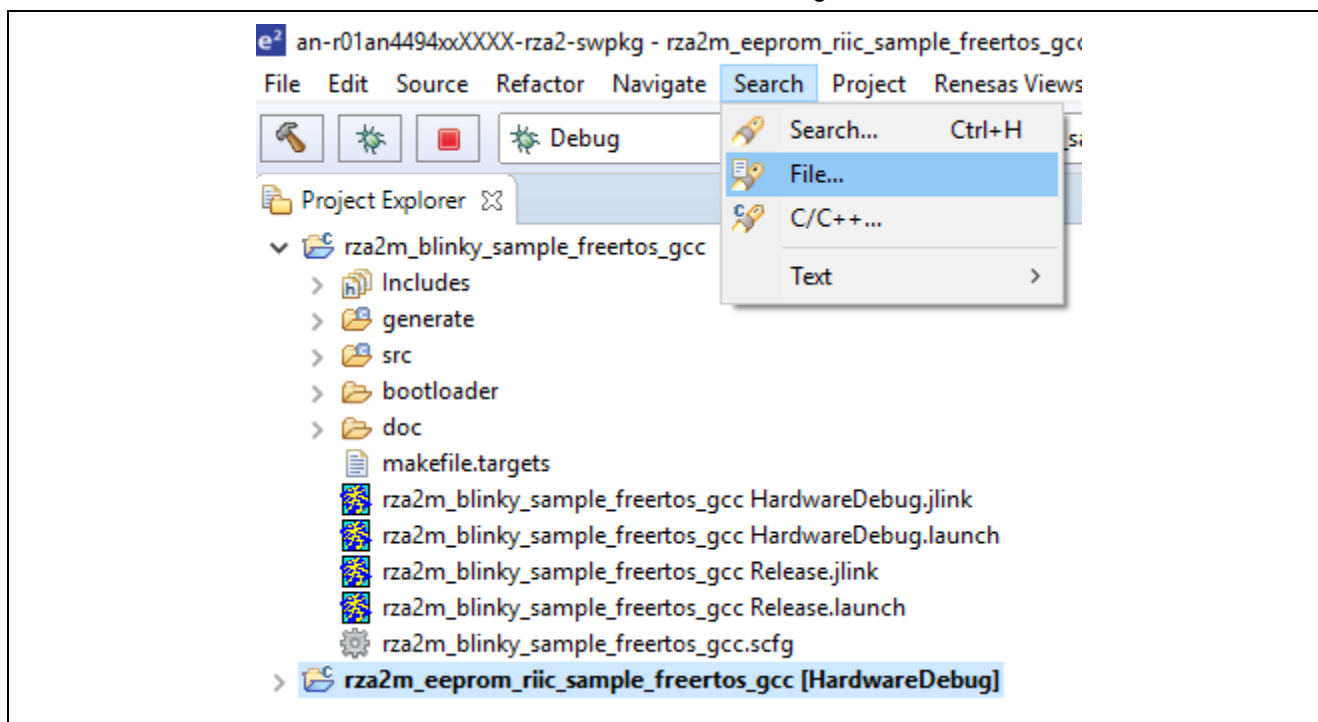


Figure 4-5 Searching the file that includes “r_riic_drv_api.h” <1>

- Input “r_riic_drv_api.h” in “Containing text:” box, and press “Search” button. This is shown in Figure 4-6.

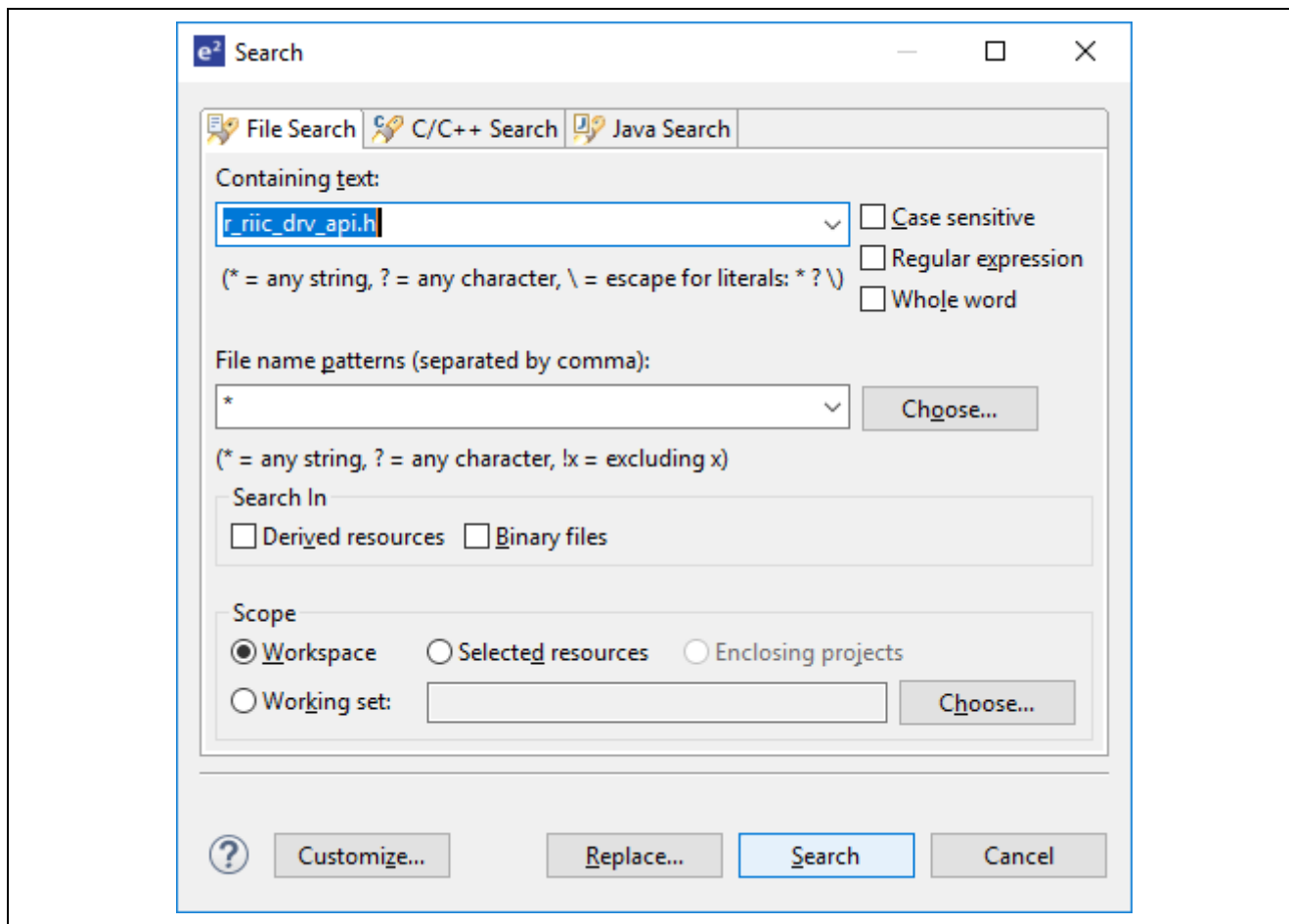


Figure 4-6 Searching the file that includes “r_riic_drv_api.h” <2>

- The result shows that “src\renesas\application\r_eeeprom_sample.c” in “rza2m_eeeprom_riic_sample_freertos_gcc” project includes “r_riic_drv_api.h.” Open “r_eeeprom_sample.c” by double-clicking. This is shown in Figure 4-7.

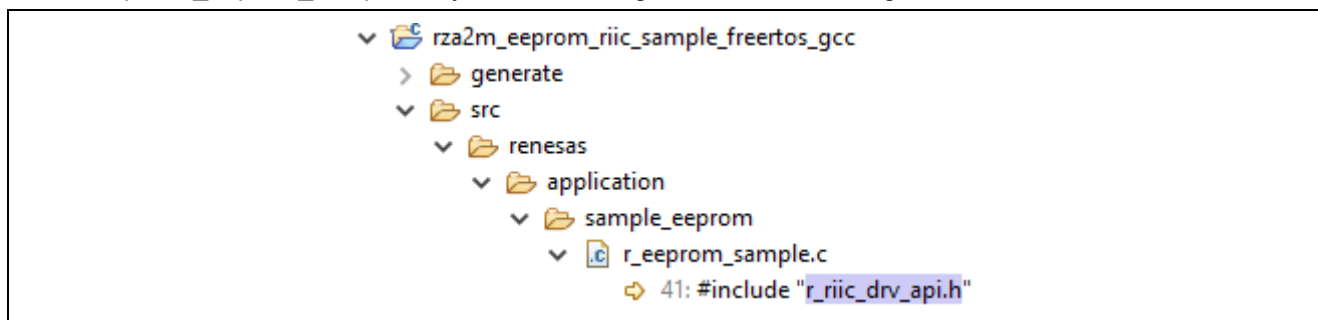


Figure 4-7 Searching the file that includes “r_riic_drv_api.h” <3>

- The following examples are implemented in “r_eeeprom_sample.c”, and they are shown in Figure 4-8.
- Initialize driver by open function.
 - Un-initialize driver by close function.
 - Read and write by control function.

```
/* RIIC driver handle */
static int_t gs_handle;

⊕ Exported global variables and functions (to be accessed by other modules)

⊕ * Function Name: sample_riic_eeeprom_init
⊖ int_t sample_riic_eeeprom_init(void)
{
    int_t ret = NO_ERROR;

    gs_handle = open(RIIC_CH_EEPROM_RZA2M, O_RDWR);

    ⊖ if (0 > gs_handle)
    {
        ret = ERROR_FAILURE;
    }

    /* Successful initialisation */
    return ret;
}
```

```
⊖ int_t sample_riic_eeeprom_uninit(void)
{
    close(gs_handle);

    return NO_ERROR;
}
```

```
/* EEPROM page write */
st_r_drv_riic_transfer_t i2c_write;
i2c_write.device_address = EEPROM_DEV_ADDRESS;
i2c_write.sub_address_type = RIIC_SUB_ADDR_WIDTH_16_BITS;
i2c_write.sub_address = write_addr;
i2c_write.number_of_bytes = write_size;
i2c_write.p_data_buffer = p_data;

drv_ret = control(gs_handle, CTL_RIIC_WRITE, &i2c_write);
```

```
st_r_drv_riic_transfer_t i2c_dummy_read;
i2c_dummy_read.device_address = EEPROM_DEV_ADDRESS;
i2c_dummy_read.sub_address_type = RIIC_SUB_ADDR_WIDTH_16_BITS;
i2c_dummy_read.sub_address = dummy_addr;
i2c_dummy_read.number_of_bytes = dummy_size;
i2c_dummy_read.p_data_buffer = dummy_buf;

error_cnt = 0;
while (1)
{
    /* wait for ACK from EEPROM */
    /* Since NACK is returned until writing is completed, an error occurs */
    drv_ret = control(gs_handle, CTL_RIIC_READ, &i2c_dummy_read);
    if (drv_err == ERROR_FAILURE)
    {
        error_cnt++;
        continue;
    }
}
```

Figure 4-8 Examples of using driver implemented in “r_riic_drv_api.h”

- Modify main.c in “rza2m_blinky_sample_freertos_gcc” project, by referring “r_eeeprom_sample.c”.
Figure 4-9 shows the operations added to main.c in rza2m_blinky_sample_freertos_gcc.
Figure 4-10 shows the debugging output by running the modified project.

```

int_t gs_handle;

/* Open RIIC driver */
gs_handle = open(RIIC_CH_EEPROM_RZA2M, O_RDWR);

if (0 <= gs_handle)
{
    int_t drv_ret = DRV_SUCCESS;
    /* EEPROM page0 write 1byte, 0x5a */
    st_r_drv_riic_transfer_t i2c_write;
    const uint8_t data = 0x5a;
    i2c_write.device_address = EEPROM_DEV_ADDRESS;
    i2c_write.sub_address_type = RIIC_SUB_ADDR_WIDTH_16_BITS;
    i2c_write.sub_address = 0;
    i2c_write.number_of_bytes = 1;
    i2c_write.p_data_buffer = &data;

    drv_ret = control(gs_handle, CTL_RIIC_WRITE, &i2c_write);
    if (DRV_ERROR != drv_ret)
    {
        st_r_drv_riic_transfer_t i2c_read;
        uint8_t result=0;
        /* wait 100ms */
        R_OS_TaskSleep(100);
        /* EEPROM page0 read 1byte */
        i2c_read.device_address = EEPROM_DEV_ADDRESS;
        i2c_read.sub_address_type = RIIC_SUB_ADDR_WIDTH_16_BITS;
        i2c_read.sub_address = 0;
        i2c_read.number_of_bytes = 1;
        i2c_read.p_data_buffer = &result;

        drv_ret = control(gs_handle, CTL_RIIC_READ, &i2c_read);
        if (DRV_ERROR != drv_ret)
        {
            printf ("EEPROM success data = %02X\n",result);
        }
    }
    /* Close RIIC driver */
    close(gs_handle);
}

```

Figure 4-9 Added code to “main.c” in “rza2m_blinky_sample_freertos_gcc” project

```

RZ/A2M blinky_sample for GCC Ver. 3.1
Copyright (C) 2018 Renesas Electronics Corporation. All rights reserved.
Build Info Date Mar 19 2019 at 21:49:31
FreeRTOS OS Abstraction Version 3.0

SAMPLE> EEPROM success data = 5A

```

Figure 4-10 Output by running modified “rza2m_blinky_sample_freertos_gcc” project.

5. How to create your own FreeRTOS project

When you create a new project that uses FreeRTOS, select the following creation method according to your purpose.

When using Amazon AWS functions: Go to section 5.1.

When not using the Amazon AWS function: Go to section 5.2.

5.1 How to create FreeRTOS project that uses Amazon AWS

Create a project using the new C/C ++ project creation function of e²studio.

For details of the procedure, refer to the following document.

<https://www.renesas.com/document/qsg/getting-started-renesas-rza2m-evaluation-board-kit-0?language=en&r=1305306>

5.2 How to create FreeRTOS project without Amazon AWS

The new project created by e²studio always includes AWS functions.

If you only want to use the FreeRTOS kernel, import the existing sample project.

We recommend using the following projects as a base.

rza2m_blinky_sample_freertos_gcc.zip included in RZ/A2M Simple Applications Package(R01AN4494).

6. FreeRTOS awareness function

In this section it is shown that FreeRTOS awareness function of e2 studio.

This function supports displaying the list and the status of generated tasks, queues, and timer during break.

6.1 Adding FreeRTOS awareness function to e² studio

6.1.1 In the case of new e² studio installation or upgrading e² studio

— Launch e² studio installer. If e² studio has already been installed, select “Upgrade” or “Install”. This is shown in Figure 6-1.

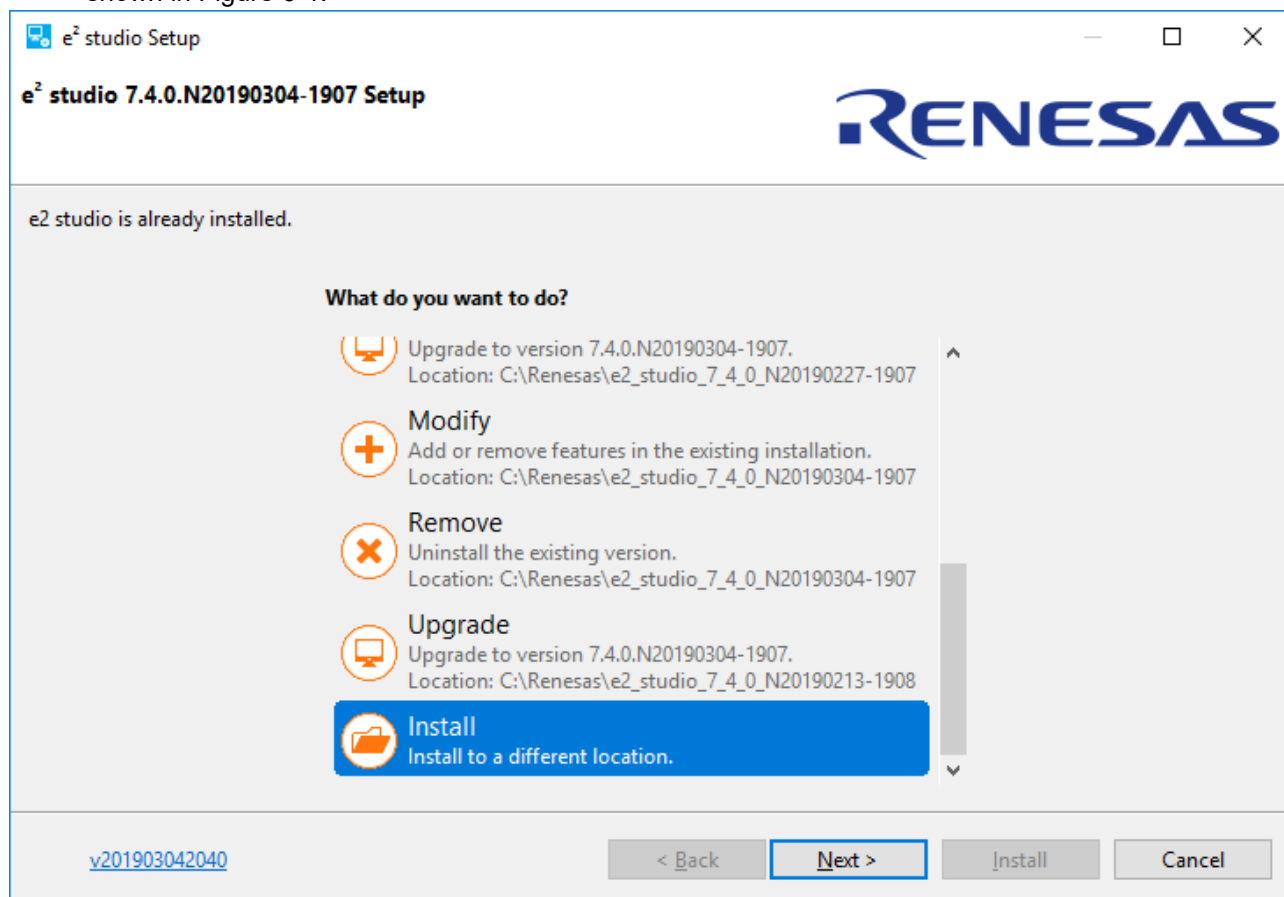


Figure 6-1 Adding FreeRTOS awareness function (new e² studio installation) 1

— Check “RZ” at “Device Families” stage of install wizard. This is shown in Figure 6-2.

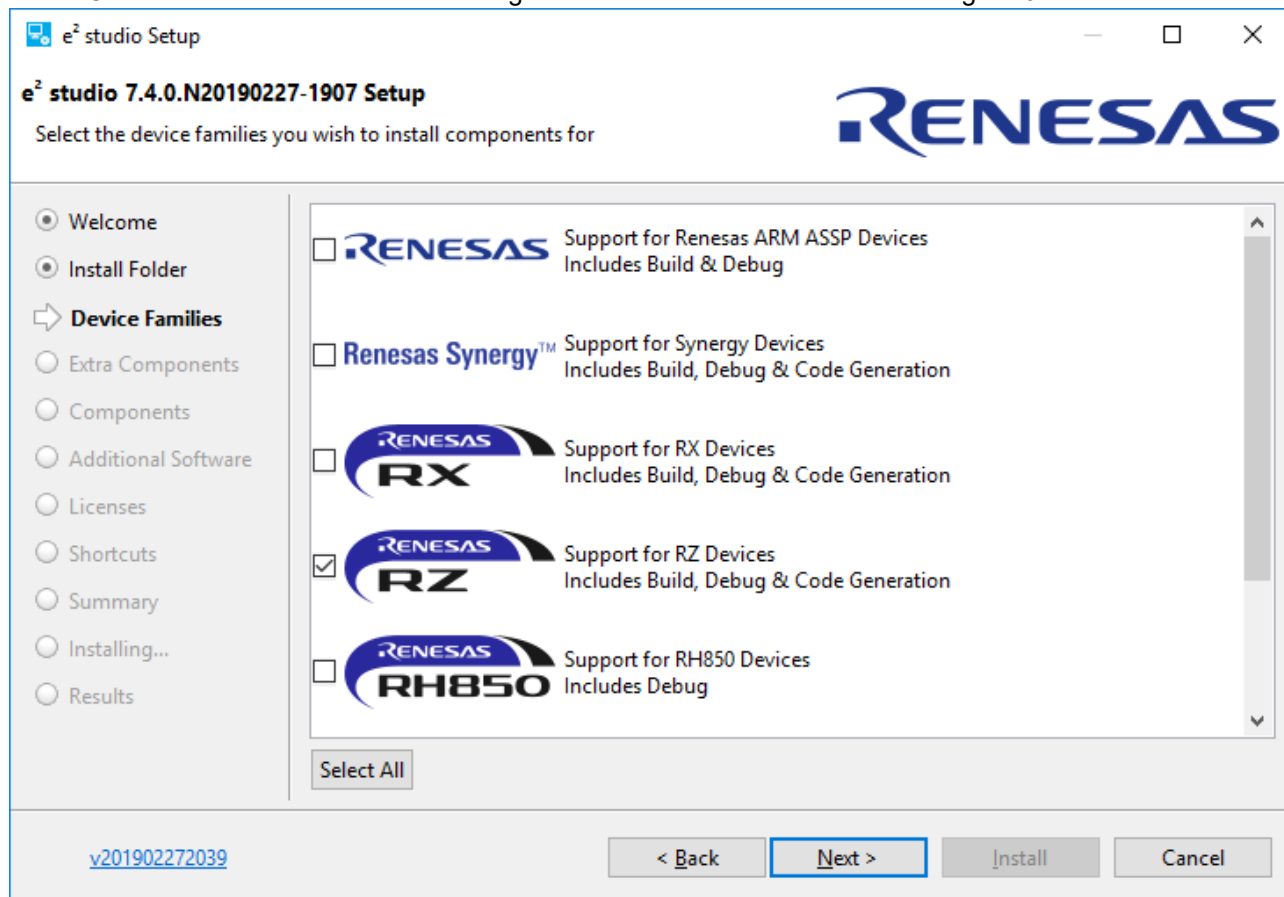


Figure 6-2 Adding FreeRTOS awareness function (new e² studio installation) 2

— Check “RTOS” at “Extra Components” stage of install wizard. This is shown in Figure 6-3.

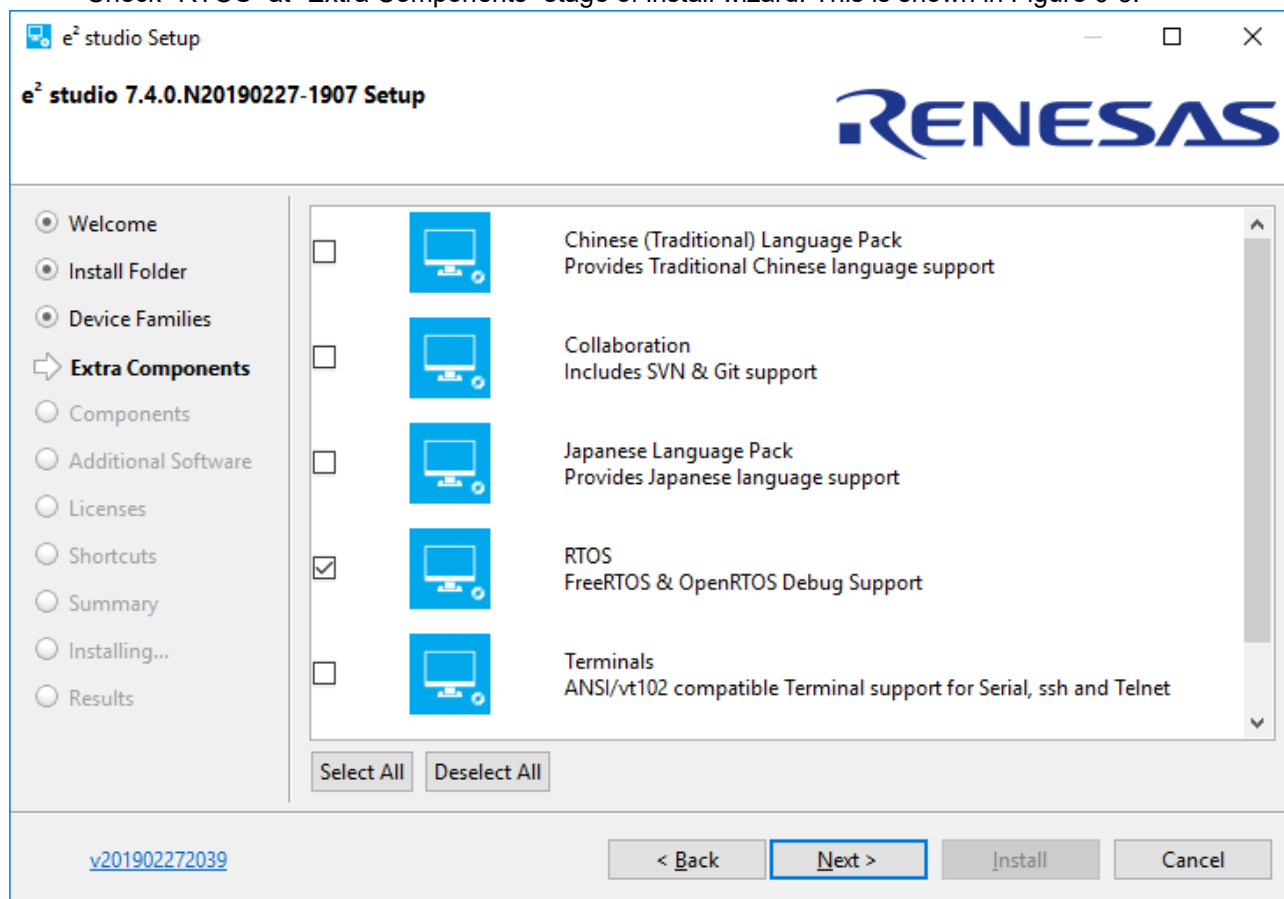


Figure 6-3 Adding FreeRTOS awareness function (new e² studio installation) 3

- Check “GCC ARM Embedded 6 2017q2” and “LibGen for GCC ARM Embedded” at “Additional Software” stage of install wizard. This is shown in Figure 6-4.

By setting up to here, FreeRTOS awareness function will be enabled.

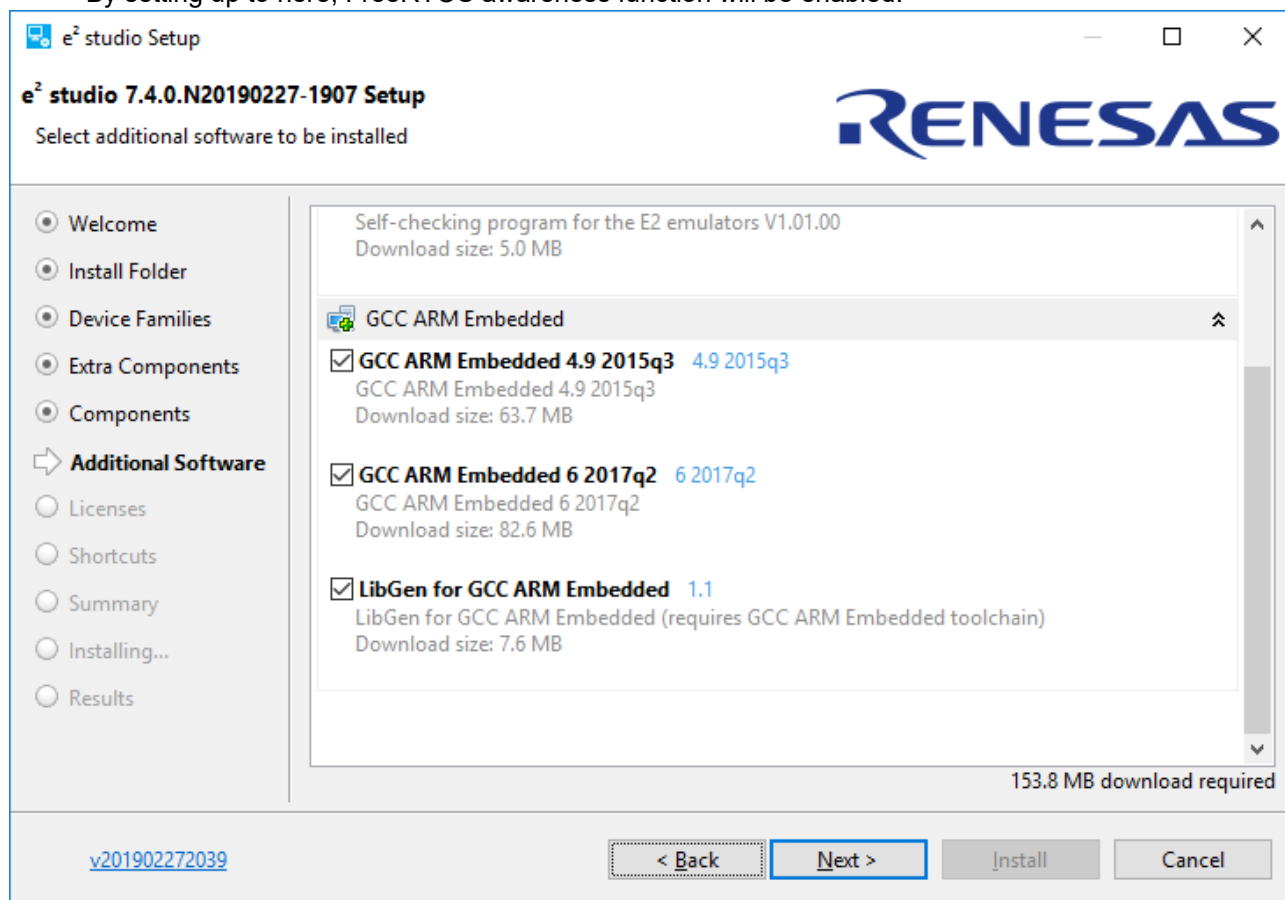


Figure 6-4 Adding FreeRTOS awareness function (new e² studio installation) 4

6.1.2 In the case that e² studio has already been installed

— Launch e² studio installer. And select “Modify”. This is shown in Figure 6-5.

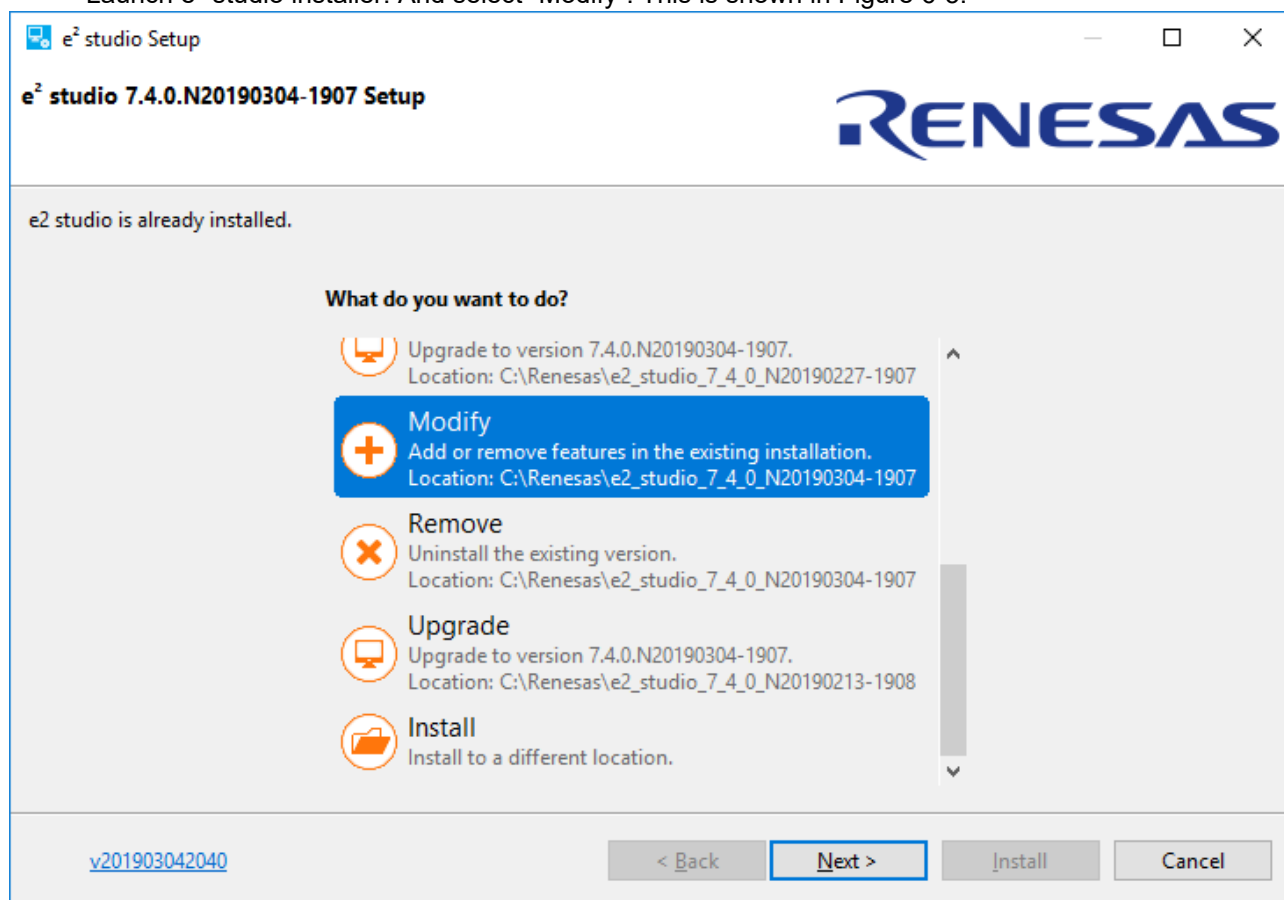


Figure 6-5 Adding FreeRTOS awareness function (to existing e² studio) 1

- Check “Renesas RTOS Debug Views” at “Components” stage of install wizard. This is shown in Figure 6-6.

By setting up to here, FreeRTOS awareness function will be enabled.

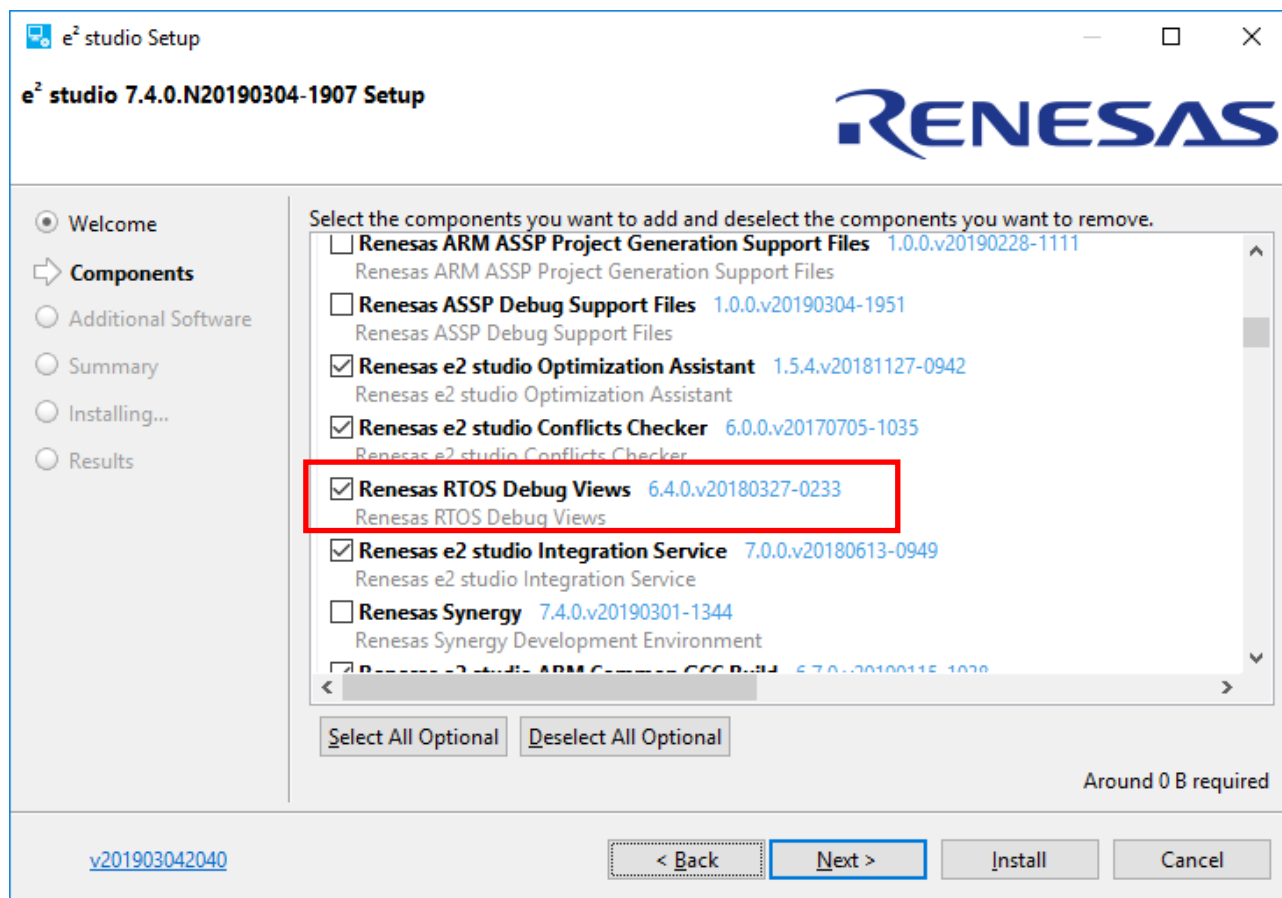


Figure 6-6 Adding FreeRTOS awareness function (to existing e² studio) 2

6.2 How to launch FreeRTOS awareness function

1. Download the program using FreeRTOS to your board.
2. Run the downloaded program.
3. Suspend (break) the running program.
4. Select “Window” menu – “Show view” – “Other”. This is shown in Figure 6-7.

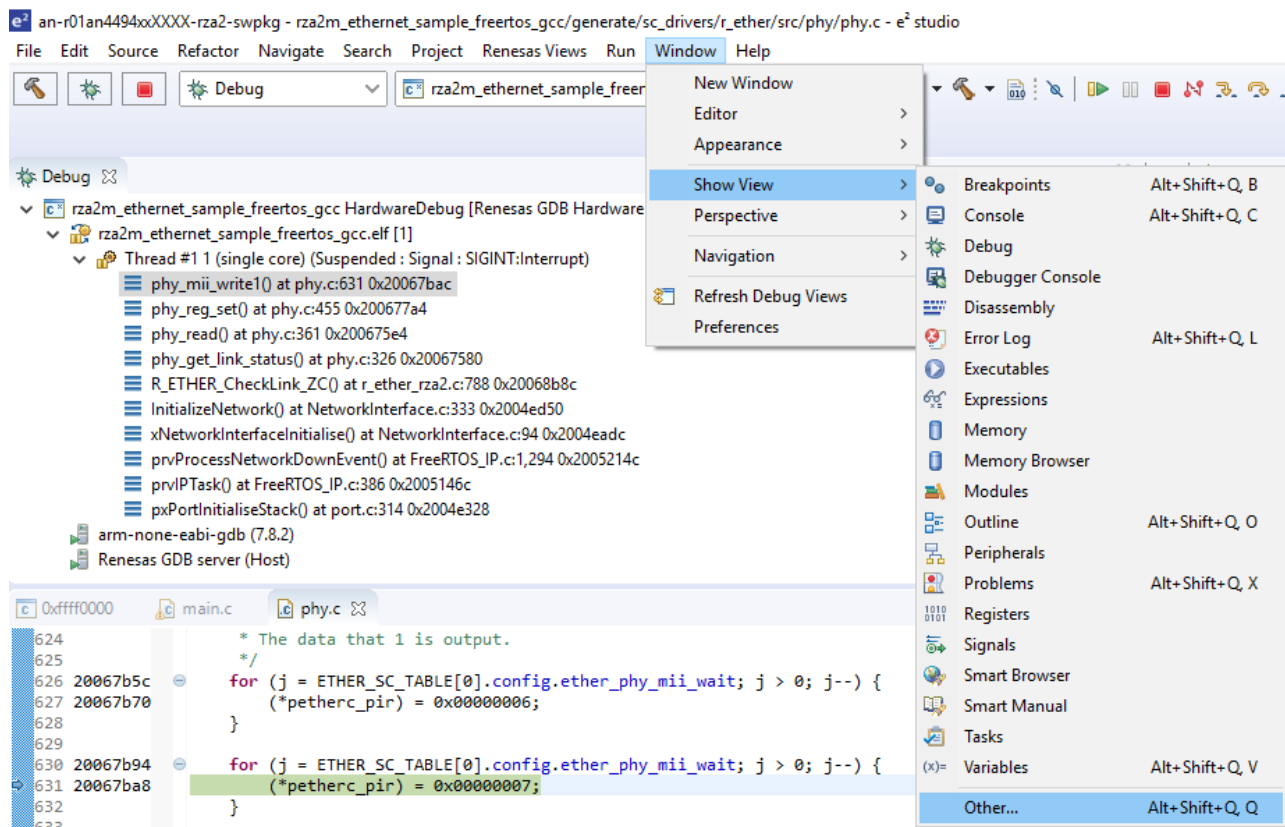


Figure 6-7 Launching FreeRTOS awareness function, step 4

5. Select "Queue Table" under "OpenRTOS Viewer". And Press "Open". This is shown in Figure 6-8.
6. After the same procedure, select "Task Table" under "OpenRTOS Viewer". And Press "Open".
7. After the same procedure, select "Timer Table" under "OpenRTOS Viewer". And Press "Open".

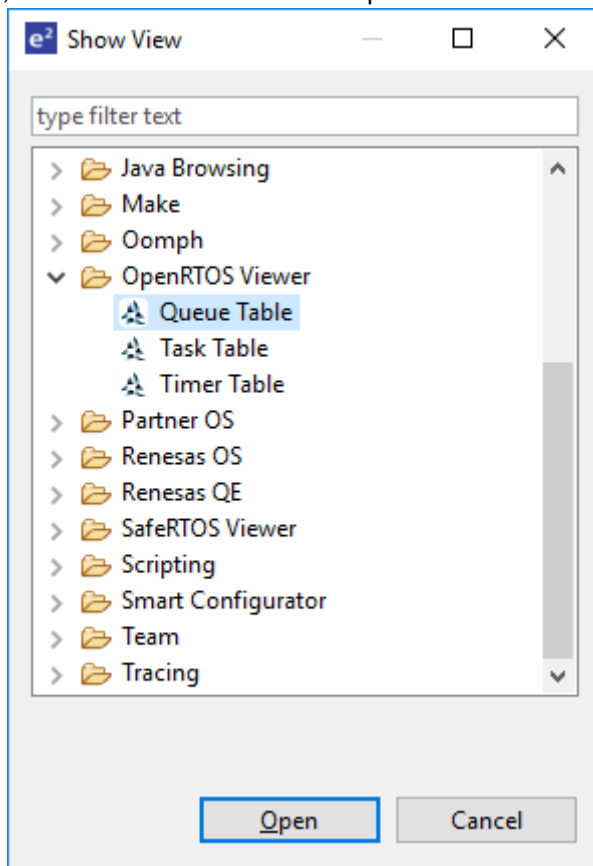


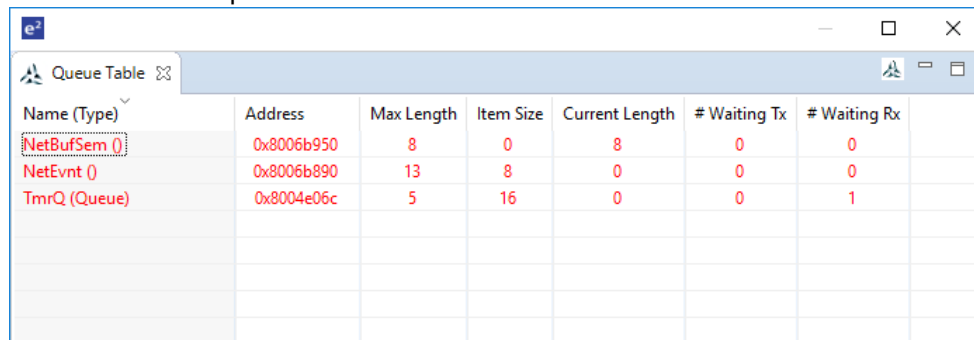
Figure 6-8 Launching FreeRTOS awareness function, step 5,6, and 7

8. Status of FreeRTOS will be shown.

Figure 6-9 shows the example of Queue Table view.

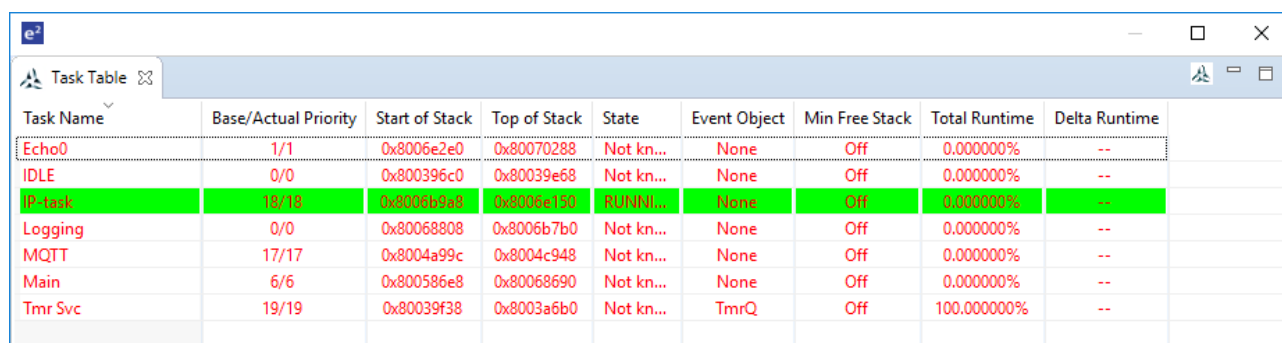
Figure 6-10 shows the example of Task Table view.

Figure 6-11 shows the example of Timer Table view.



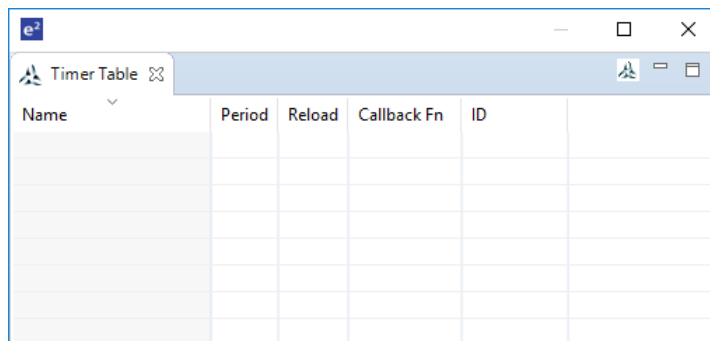
Name (Type)	Address	Max Length	Item Size	Current Length	# Waiting Tx	# Waiting Rx
NetBufSem ()	0x8006b950	8	0	8	0	0
NetEvt ()	0x8006b890	13	8	0	0	0
TmrQ (Queue)	0x8004e06c	5	16	0	0	1

Figure 6-9 FreeRTOS awareness function (Queue Table)



Task Name	Base/Actual Priority	Start of Stack	Top of Stack	State	Event Object	Min Free Stack	Total Runtime	Delta Runtime
Echo0	1/1	0x8006e2e0	0x80070288	Not kn...	None	Off	0.000000%	--
IDLE	0/0	0x800396c0	0x80039e68	Not kn...	None	Off	0.000000%	--
IP-task	18/18	0x8006b9a8	0x8006e150	RUNNI...	None	Off	0.000000%	--
Logging	0/0	0x80068808	0x8006b7b0	Not kn...	None	Off	0.000000%	--
MQTT	17/17	0x8004a99c	0x8004c948	Not kn...	None	Off	0.000000%	--
Main	6/6	0x800586e8	0x80068690	Not kn...	None	Off	0.000000%	--
Tmr Svc	19/19	0x80039f38	0x8003a6b0	Not kn...	TmrQ	Off	100.000000%	--

Figure 6-10 FreeRTOS awareness function (Task Table)



Name	Period	Reload	Callback Fn	ID

Figure 6-11 FreeRTOS awareness function (Timer Table)

7. How to use e²studio virtual console

e²studio has a virtual console function.

You can redirect standard I/O to virtual console by making certain settings in your project.

This section explains the virtual console usage conditions, setup procedures, and notes.

7.1 Requirements for using the virtual console

The virtual console can be used when the following conditions are met.

- e²studio version 7.7 or later
- Target project is OSless

7.2 Limitations of virtual console

The virtual console has the following restrictions. It does not completely replace the existing standard I/O.

Please use the virtual console after understanding the restrictions.

- The communication speed is very slow, so it is not suitable for a large amount of input/output.
- The program stops completely while waiting for input of `getc` or `scanf`. (No interrupt occurs)
- Buffering operations (`fflush`, `setvbuf`, etc.) for the virtual console are prohibited.
- The program cannot run standalone if virtual console is enabled. Create the final project with the virtual console disabled.

7.3 How to enable virtual console for your project

This section describes how to enable virtual console for the rza2m_blinky_sample_osless_gcc project bundled in RZ/A2M Simple Applications Package(R01AN4494)

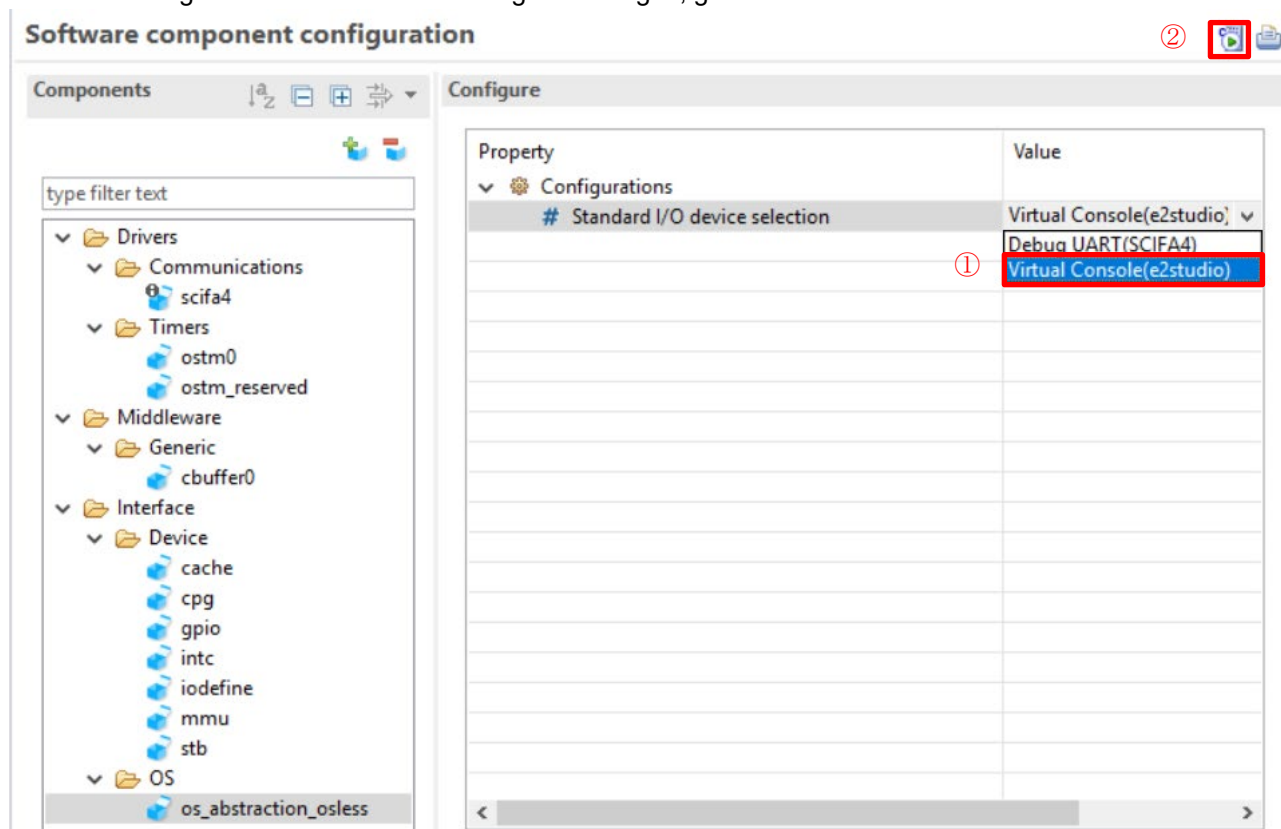
Summary of steps

1. Enable Virtual Console on Smart Configurator
2. Add the following two modifications to r_devlink_wrapper_cfg.h.
 - Add #include "r_os_abstraction_api.h".
 - Add standard I/O device description for virtual console.
3. Remove buffer operations for standard I/O from application source.
 - Remove all of setvbuf.
 - Remove all of fflush.
4. Build the project.
5. Check svc_handler address in Map file.
6. Register the svc_handler address in the debug connection setting file.
7. Download the program.
8. Open virtual console view.
9. Run the program.

The detailed procedure is described below.

1. Enable Virtual Console on Smart Configurator

In the Components tab of SmartConfigurator, select os_abstraction_osless and change the properties as shown in the figure below. And after making the changes, generate the code.



2. Register standard input/output file name for virtual console.

Edit `generate/configuration/r_devlink_wrapper_cfg.h`.

First, change lines 40 and 41 to the following.

```
/* Modified by user, drivers that are not under the control of sc added here */
#include "r_os_abstraction_api.h"
/* End of user modification */
```

Next, change lines 62-71 to the following.

```
#if (R_OS_ENABLE_VIRTUAL_CONSOLE == 0)
    /** SCIFA Channel 4 Driver added by USER */
    {"stdin", (const st_r_driver_t*)&g_scifa_driver, R_SC0},

    /** SCIFA Channel 4 Driver added by USER */
    {"stdout", (const st_r_driver_t*)&g_scifa_driver, R_SC0},

    /** SCIFA Channel 4 Driver added by USER */
    {"stderr", (const st_r_driver_t*)&g_scifa_driver, R_SC0},
#else
    {"stdin", (const st_r_driver_t*)&g_stdio_driver, R_SC0},
    {"stdout", (const st_r_driver_t*)&g_stdio_driver, R_SC0},
    {"stderr", (const st_r_driver_t*)&g_stdio_driver, R_SC0},
#endif
```

3. Removal of buffer operations for standard I/O.

First, edit `src/user_prog/main.c` and comment out calls to `setvbuf`.

Then edit `src/renesas/application/console/console.c` and comment out all calls to `fflush` and `setvbuf`.

4. Build the project.

Hereafter, it is assumed that you have built the HardwareDebug configuration.

5. Check `svc_handler` address in Map file.

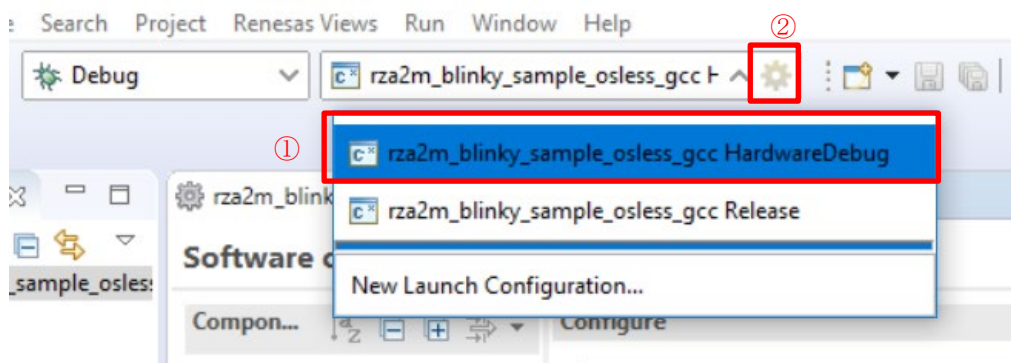
Open `Hardwaredebug\rza2m_blinky_sample_osless_gcc.map` with an editor and check the address of `svc_handler`.

Here, the explanation is continued assuming that the address is `0x20011460` as shown below.

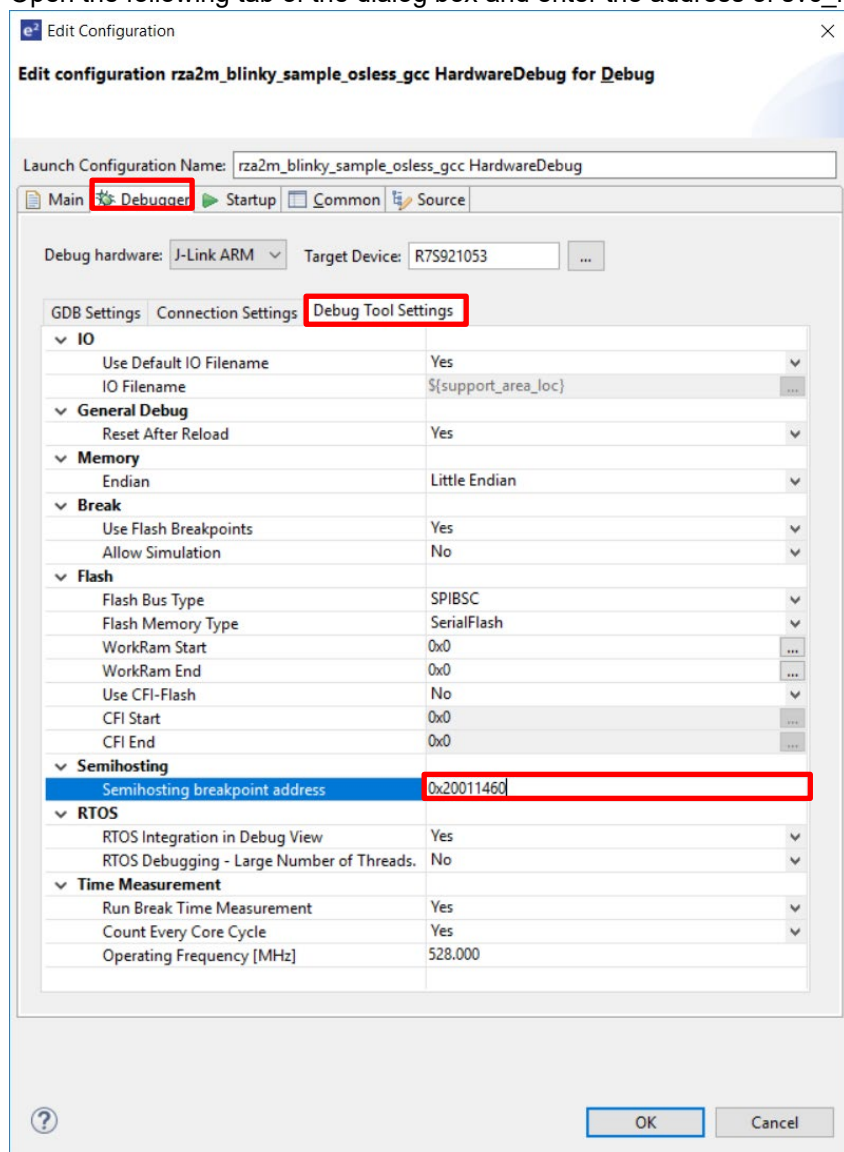
```
RESET_HANDLER 0x20011280 0x1f0 ./generate/compiler/asm/reset_handler.o
    0x20011280      reset_handler
    0x2001145c      undefined_handler
    0x20011460      svc_handler
    0x20011464      prefetch_handler
    0x20011468      abort_handler
    0x2001146c      reserved_handler
```

6. Register the svc_handler address in the debug connection setting file.
From the bar at the top of e2 studio, select Edit Debugger Connection Settings for HardwareDebug.

blinky_sample_osless_gcc/rza2m_blinky_sample_osless_gcc.scfg - e² studio

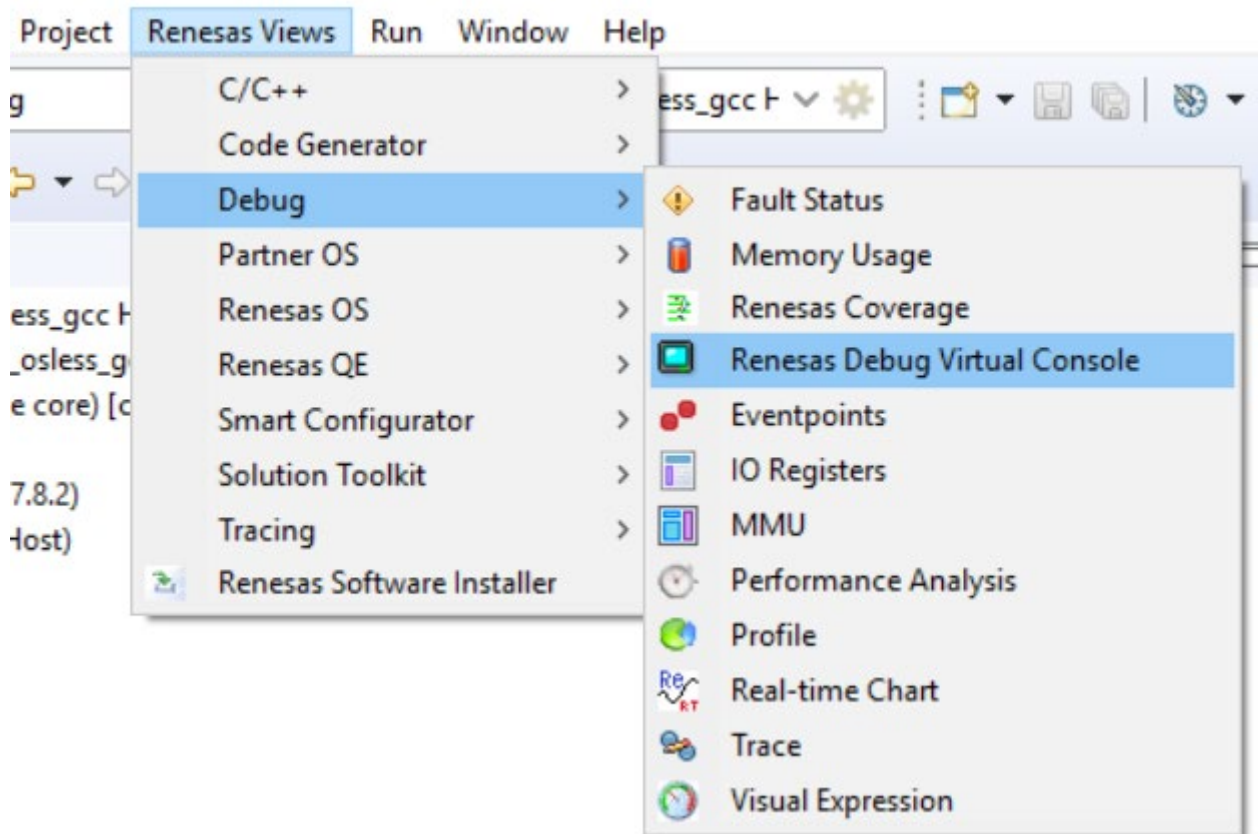


Open the following tab of the dialog box and enter the address of svc_handler

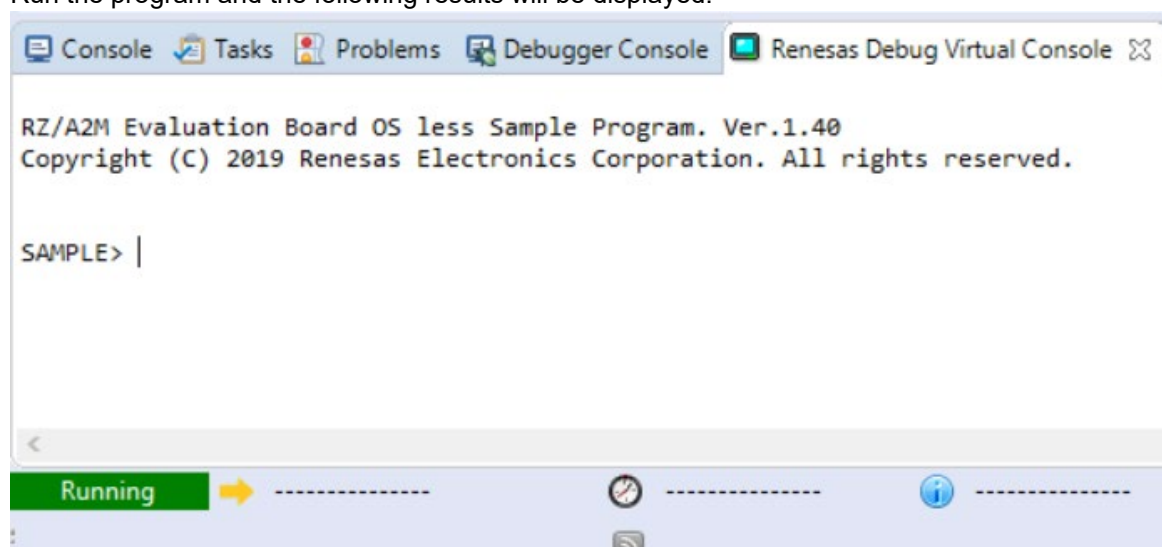


7. Download the program.
Start debugging in the environment set in the previous section.
8. Open virtual console view.
Select the following view from the Renesas Views menu of e²studio.

- e² studio



9. Run the program.
Run the program and the following results will be displayed.



Note: The blinking LED has stopped because the program has stopped waiting for console input.

8. Notes on updating driver components

If you want to apply the latest driver to an existing project, there are some tasks in addition to updating the driver in SmartConfigurator that require manual changes to the project.

Please update your project by following the steps below.

8.1 r_iodefine

Older projects do not contain r_iodefine components. In the project that does not include r_iodefine, the header files are placed directly.

If you add r_iodefine to such a project, manually delete the old header files below.

- generate\system\inc\iobitmask.h
- generate\system\inc\iodefine.h
- generate\system\inc\iobitmasks folder
- generate\system\inc\iodefines folder

8.2 r_os_abstraction

8.2.1 Update from 3.04 or earlier

Some compiler-dependent files have been moved to r_os_abstraction for virtual console support.

Please delete the following files.

- generate\compiler\inc\swi.h
- generate\compiler\init_exit.c
- generate\compiler\init_kill.c
- generate\compiler\init\syscalls.c

If the current version is 3.03 or earlier, see also the next section.

8.2.2 Update from 3.03 or earlier

Add the following definition to generate\compiler\inc\r_compiler_abstraction_api.h.

- #define R_COMPILER_WEAK __attribute__((weak))

If you are using r_os_abstraction_freertos, you also need to make the following changes.

The heap implementation has moved from the FreeRTOS folder to r_os_abstraction_freertos.

Please delete the following file.

- <freertos folder>\portable\memmang\heap5_renesas.c

9. Support

Online technical support and information is available at <https://www.renesas.com>

Technical Contact Details

America: techsupport.america@renesas.com

Europe: <https://www.renesas.com/eu/en/support/contact.html>

Japan & global: <https://www.renesas.com/support/contact.html>

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Dec. 28, 2018	—	First edition issued
1.01	Apr. 15, 2019	11	Added section 4.1, "Components and related sample projects"
		13	Added section 4.3, "How to integrate the new component"
		19	Added section 5, "How to create your own FreeRTOS project"
1.02	May. 17, 2019	10	Added the following reference document. • RZ/A2M Smart Configurator User's Guide
1.03	Sep. 30, 2019	2	Added Install guide of e2 studio.
		2	Changed e2 studio version to 7.5.
		11	Added OS-less version of sdhi_fat sample.
		11	Added adc sample and r_adc driver.
		11	Added usbf_cdc sample. And added r_usbf_basic and r_usbf_cdc drivers.
		11	Added wifi_pmod_esp32 sample.
1.04	Dec. 17, 2019	2	Changed e2 studio version to 7.6.
		7	Changed program download process
		11	Added rtc sample and r_rtc driver.
1.05	Mar. 31, 2020	2	Changed e2 studio version to 7.7.
		11	Added fw_update_boot sample and fw_update_sample sample. Added gpt-pwm sample and r_gpt driver. Added r_hyperbus driver. Added touch_panel sample.
1.06	Jun. 30, 2020	2	Changed e2 studio version to 7.8.
		19	Added section 5, "How to create your own FreeRTOS project"
		29	Added section 7, "How to use e2studio virtual console"
		34	Added section 8, "Notes on updating driver components"
1.07	Sep.30, 2020	2	Changed e2 studio version to 2020-07.
		11	Added ssif sample and r_ssif driver. Added graphics sample and r_rga driver. Added wifi_sxdmac sample and silex driver.
1.08	Apr.20.2021	2	Changed e2 studio version to 2021-04.

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/