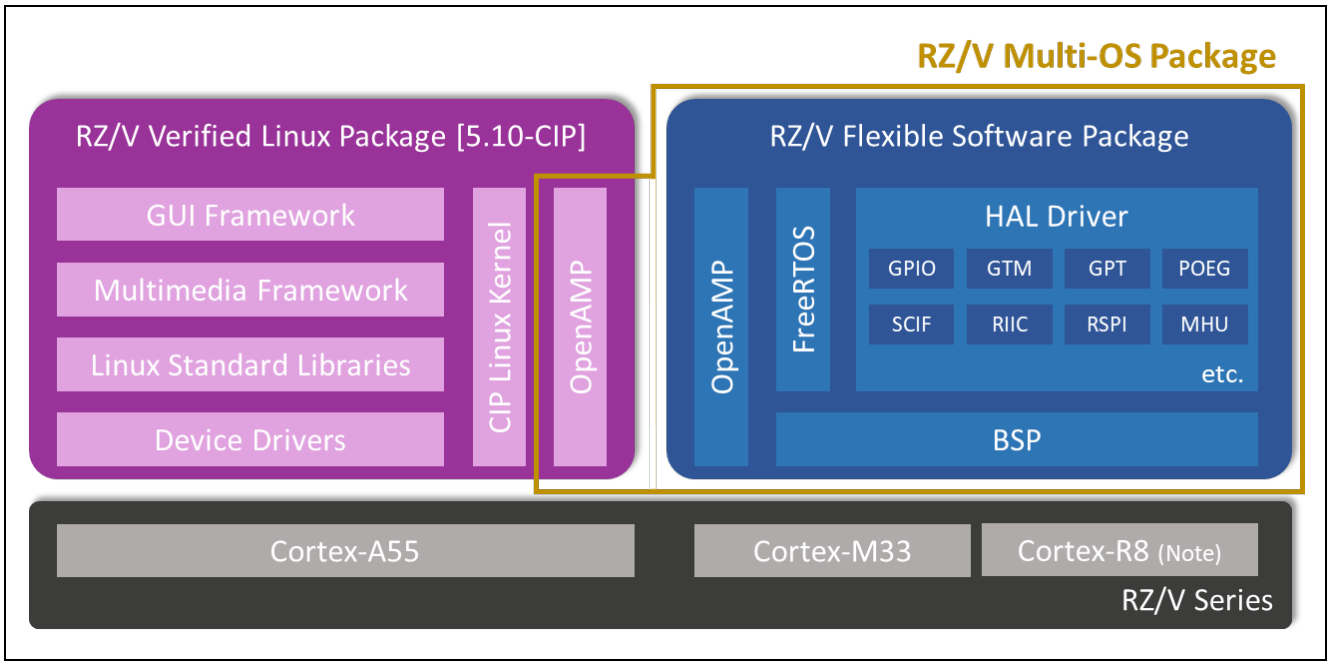
RZ/V2L, RZ/V2H

Release Note for RZ/V Multi-OS Package V2.1.0

Introduction

This software package provides user with easy way to establish multi-OS (i.e., CIP Linux running on Cortex®-A55 and FreeRTOS running on Cortex-M33 and/or Cortex-R8) environment and sample program showing how to implement Inter-Processor Communication between those CPU cores.

This package consists of RZ/V Flexible Software Package (hereinafter referred to as RZ/V FSP) and Multi-OS Feature Package for RZ/V Verified Linux Package (hereinafter referred to as RZ/V VLP).



Note: Cortex-R8 is available on RZ/V2H only.

Here are brief descriptions of each component of RZ/V Multi-OS Package:

- RZ/V FSP
  The software package consisting of production ready peripheral drivers, FreeRTOS and portable middleware stacks and the best in-case HAL drivers with low memory footprint.
- OpenAMP
  The framework including the software components needed for Asymmetric Multiprocessing (AMP) systems such as Inter-Processor Communication.

Target Device

RZ/V2L, RZ/V2H

Contents

1. Specifications

Table 1-1 lists the on-chip peripheral modules to be used in this application.

Table 1-1. /Peripheral module to be used

Peripheral module	Usage
Message Handling Unit (MHU)	Configures Inter-Processor Interrupt.
Serial Communications Interface with FIFO (SCIFA)	Performs standard serial communications sending and receiving console messages.
Interrupt controller (INTC)	Configures interrupt settings; the processor will receive interrupts during buffered serial communications, and in the MHU module when Inter-Processor Interrupt is fired.
General Purpose Input Output (GPIO)	Configures I/O lines used by serial communications.
General Timer (GTM)	Configures the tick for FreeRTOS.

2. Verified Operation Conditions

Table 2-1. Verified Operating Conditions

Item	Contents
Microprocessor used	RZ/V2L, RZ/V2H
Integrated Development Environment	e^2 studio 2024-07
Toolchain	GNU Arm Embedded 12.2.Rel1

3. Sample Program Setup

3.1 RZ/V2L Flexible Software Package Setup

Please refer to Getting Started with RZ/V Flexible Software Package.

3.2 OpenAMP related stuff Integration for RZ/V2L

This section describes how to integrate OpenAMP related stuff to RZ/V Verified Linux Package [5.10-CIP] (hereinafter referred to as VLP). For details, please refer to SMARC EVK of RZ/V2L Linux Start-up Guide (r01us0617ej0103-rz-v(Linux Start-up Guide RZV2L.pdf) included in VLP.

1. Follow the procedure from the beginning of **2.2 Building Images** to **(4) Add layers** of **SMARC EVK of RZ/V2L Linux Start-up Guide**.
2. Download Multi-OS Package (r01an7254ej0210-rzv-multi-os-pkg.zip) to working directory and run the commands stated below:

```
$ cd ~/rzv_vlp_<pkg ver> (Note 1)
$ unzip <Multi-OS Dir>/r01an7254ej0210-rzv-multi-os-pkg.zip (Note 2)
$ tar zxvf r01an7254ej0210-rzv-multi-os-pkg/meta-rz-features_multi-os_v2.1.0.tar.gz
```

Notes: 1. <pkg ver> stands for the version number of VLP/V (e.g., 3.0.6).
2. <Multi-OS Dir> stands for the path to the directory where Multi-OS Package is placed.

3. Add the layer for Multi-OS Package.

```
$ bitbake-layers add-layer ../meta-rz-features/meta-rz-multi-os/meta-rzv2l
```

4. Start a build as described in **(6) Start a build** of **2.2 Building Images** in **SMARC EVK of RZ/V2L Linux Start-up Guide**.

```
$ MACHINE=smarc-rzv2l bitbake core-image-<target>
```

Here, minimal, bsp, weston or qt can be specified for <target>. For details, please refer to **SMARC EVK of RZ/V2L Linux Start-up Guide**.

3.3 OpenAMP related stuff Integration for RZ/V2H

This section describes how to integrate OpenAMP related stuff to RZ/V2H AI Software Development Kit (hereinafter referred to as SDK).

1. Install the package needed for building BSP.

```
$ sudo apt-get update
$ sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib
build-essential chrpath socat cpio python python3 python3-pip python3-pexpect
xz-utils debianutils iputils-ping libsdl1.2-dev xterm p7zip-full libyaml-dev
libssl-dev bmap-tools
```

2. Configure your Git.

```
$ git config --global user.email "you@example.com"
$ git config --global user.name "Your Name"
```

3. Create a working directory at your home directory and extract AI SDK Source Code there.

```
$ mkdir ~/rzv2h_ai_sdk_<pkg_ver>
$ cd ~/rzv2h_ai_sdk_<pkg_ver>
$ cp <AI SDK download dir>/<AI SDK pkg>_linux-src.zip .
$ unzip <AI SDK download dir>/<pkg ver>_linux-src.zip
$ tar zxvf <AI SDK pkg>_linux-src/rzv2h_ai-sdk_yocto_recipe_<pkg ver>.tar.gz
```

4. Download Multi-OS Package (r01an7254ej0210-rzv-multi-os-pkg.zip) to working directory and run the commands stated below:

```
$ cd ~/rzv2h_ai_sdk_<pkg_ver>
$ unzip r01an7254ej0210-rzv-multi-os-pkg.zip
$ tar zxvf r01an7254ej0210-rzv-multi-os-pkg/meta-rz-features_multi-
os_v2.0.0.tar.gz
```

5. Initialize Build Environment

```
$ TEMPLATECONF=$PWD/meta-renesas/meta-rzv2h/docs/template/conf/ source
poky/oe-init-build-env build
```

6. Add the layers. The steps add the settings to bblayers.conf.

```
$ bitbake-layers add-layer ../meta-rz-features/meta-rz-multi-os/meta-rzv2h
```

7. Start a build by invoking the commands for building VLP. Building an image might take up to a few hours depending on the user's host system performance.

```
$ MACHINE=rzv2h-evk-ver1 bitbake core-image-<target>
```

Here, the allowable values for “target” in the command are as follows:

- minimal
- bsp
- weston

3.4 Note for Integration

On RZ/V Linux, the peripherals which are NOT used enter Module Standby Mode after Linux kernel is booted up. That means the peripherals used on CM33 side might NOT become worked implicitly. To avoid this situation, we prepare for the patch shown below:

- 0003-clk-renesas-rzv2l-Set-SCIF2-OSTM2.patch for RZ/V2L
- 0003-Set-OSTM-for-MCPU-and-RCPU.patch for RZ/V2H

This patch prevents SCIF and GTM(OSTM) used in the rpmsg sample program from entering Module Standby Mode. If you have any other peripherals which should NOT enter Module Standby Mode, please apply the modification in red to the patch as shown below:

```
drivers/clk/renesas/r9a07g044-cpg.c | 2 ++
 1 file changed, 2 insertions(+)

diff --git a/drivers/clk/renesas/r9a07g044-cpg.c
b/drivers/clk/renesas/r9a07g044-cpg.c
index e27caa075af7..9e904d2e8180 100644
--- a/drivers/clk/renesas/r9a07g044-cpg.c
+++ b/drivers/clk/renesas/r9a07g044-cpg.c
@@ -449,6 +449,9 @@ static const unsigned int r9a07g044_crit_mod_clks[]
__initconst = {
        MOD_CLK_BASE + R9A07G044_IA55_PCLK,
        MOD_CLK_BASE + R9A07G044_IA55_CLK,
        MOD_CLK_BASE + R9A07G044_DMAC_ACLK,
+        MOD_CLK_BASE + R9A07G044_SCIF2_CLK_PCK,
+        MOD_CLK_BASE + R9A07G044_OSTM2_PCLK,
+        MOD_CLK_BASE + R9A07G044_xxxx,
 };
```

The line number should be revised in accordance with the number of lines you added.

With respect to the allowable value for xxxx above, please refer to the source code below:

- RZ/V2L
  https://github.com/renesas-rz/rz_linux-cip/blob/c45e3a8129dd0ee36c51079e95962b2f85472e51/drivers/clk/renesas/r9a07g044-cpg.c#L204-L384
- RZ/V2H
  https://github.com/renesas-rz/rz_linux-cip/blob/397f22ac9c748f4635c3e9b032dcd45e58908011/drivers/clk/renesas/r9a09g057-cpg.c#L328-L795

3.5 Deployment of RZ/V2L VLP

With respect to the deployment of Linux kernel, device tree and root filesystem for RZ/V2L, please refer to SMARC EVK of RZ/V2L Linux Start-up Guide.

3.6 Deployment of RZ/V2H AI SDK

With respect to the deployment of Linux kernel, device tree and root filesystem for RZ/V2H, please refer to https://renesas-rz.github.io/rzv_ai_sdk/5.00/.

4. Sample Program Invocation on RZ/V2L SMARC EVK

4.1 Hardware setup

1. Connect J-Link to RZ/V2L SMARC EVK. For details, please refer to Getting Started with RZ/V Flexible Software Package.
2. Connect Pmod USBUART to the Pmod 1 of SMARC Carrier Board as shown below for securing the console for the program running on CM33.

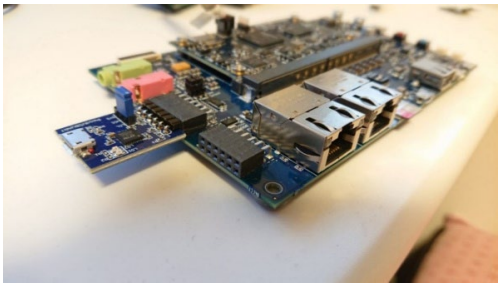


Figure 4-1. Connection between RZ/V2L SMARC EVK and Pmod USBUART

4.2 CM33 Sample Program Setup

Please carry out the following procedures for setting up demo program running on CM33.

1. Extract r01an7254ej0210-rzv-multi-os-pkg.zip on your development PC.
2. Extract rzv2l_cm33_rpmsg_demo.zip included in r01an7254ej0210-rzv-multi-os-pkg.zip.
3. Open e^2 studio 2024-07 and click File > Import.
4. Double-click General and select Existing Projects into Workspace as shown in Figure 4-2:

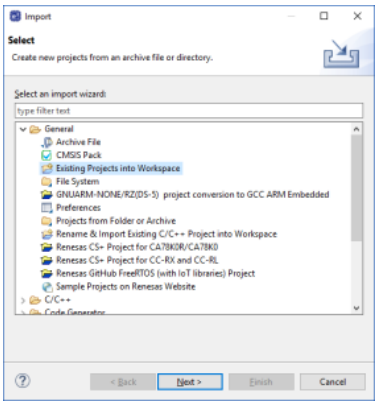


Figure 4-2. Import sample project (1)

5. Input the path to the folder rzv2l_cm33_rpmsg_demo, press Enter key and click Finish button.

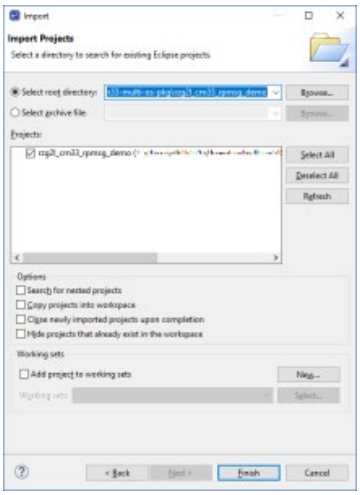


Figure 4-3. Import sample project (2)

6. (Optional) By default RPMsg channel 0 is configured to be used on CM33. If you would like to use channel 1, open the property of MainTask#0 on FSP Smart Configurator, specify 1 for Thread Context, and push **Generate Project Content** button. If **Generate Project Content** pop-up is shown, click **Proceed** to reflect the change to the source code.

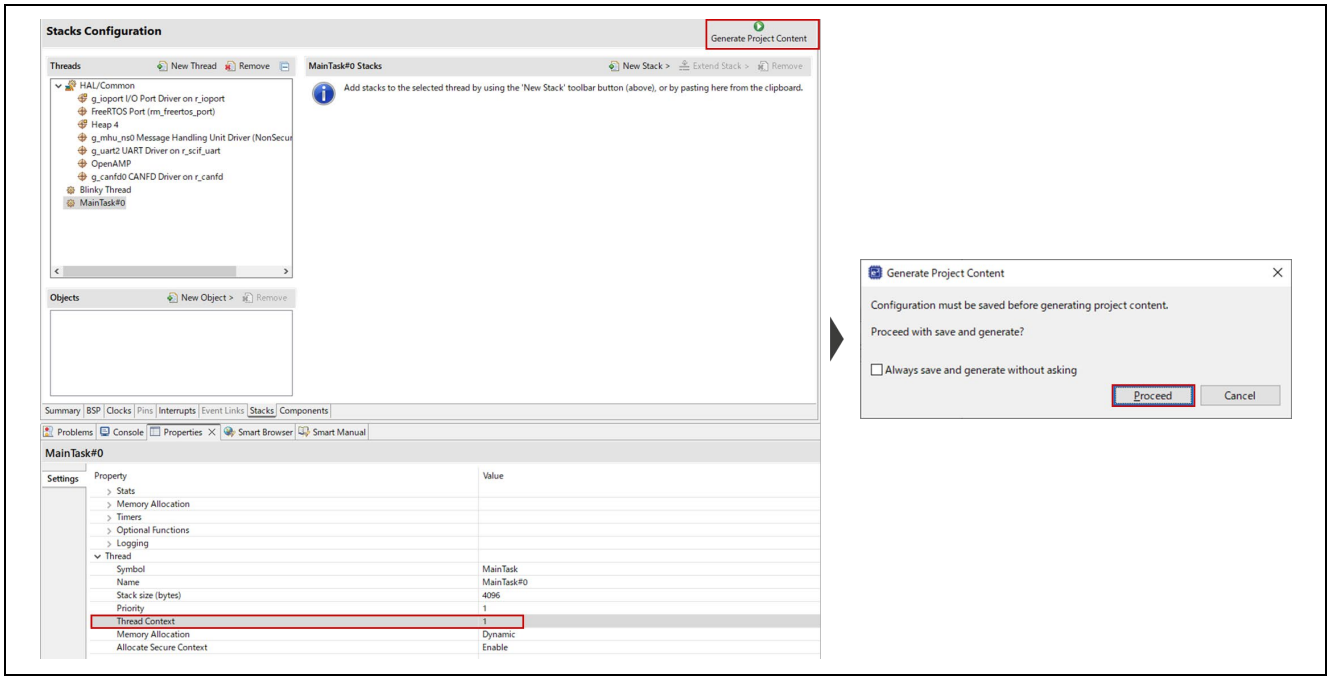


Figure 4-3. RPMsg channel setting

7. Build the project from Choose Project > Build Project.
8. If the build is successfully completed, the following files should be generated in Debug and/or Release folder in accordance with the active Build Configuration.
    - rzv2l_cm33_rpmsg_demo.elf
    - rzv2l_cm33_rpmsg_demo_non_secure_code.bin
    - rzv2l_cm33_rpmsg_demo_non_secure_vector.bin
    - rzv2l_cm33_rpmsg_demo_secure_code.bin
    - rzv2l_cm33_rpmsg_demo_secure_vector.bin

4.3 CM33 Sample Program Invocation

4.3.1 CM33 Sample Program Invocation with Segger J-Link

You need to follow the following steps to invocate CM33 sample program with Segger J-Link.

1. Choose rzv2l_cm33_rpmsg_demo Debug_Flat or rzv2l_cm33_rpmsg_demo Release_Flat from the drop-down list indicated by a red arrow in Figure 4-4.

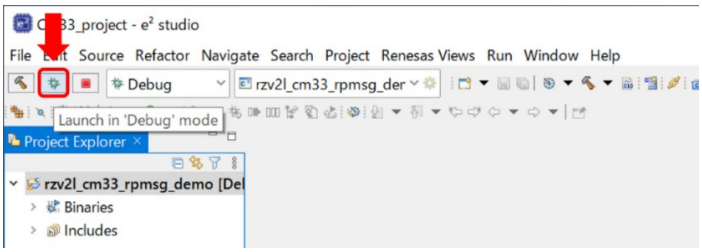


Figure 4-4. Select of Debug Configuration

2. Click the debug button indicated by a red arrow in Figure 4-5.

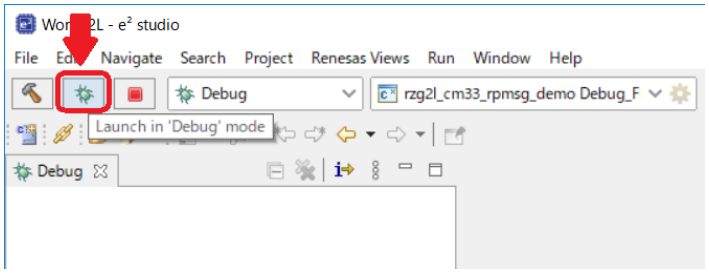


Figure 4-5. Debug Function Launch

If “Confirmation Perspective Switch” window below appears, please press “Switch” to go ahead.

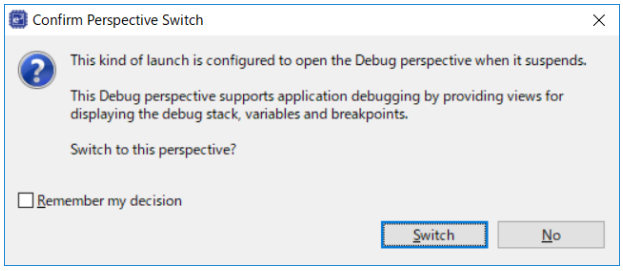


Figure 4-6. Confirmation window to open the Debug perspective

3. When the debug perspective is opened, Program Counter (PC) should be located at the top of Warm_Reset_S function. Then, you need to press the button indicated by a red arrow in Figure 4-7.

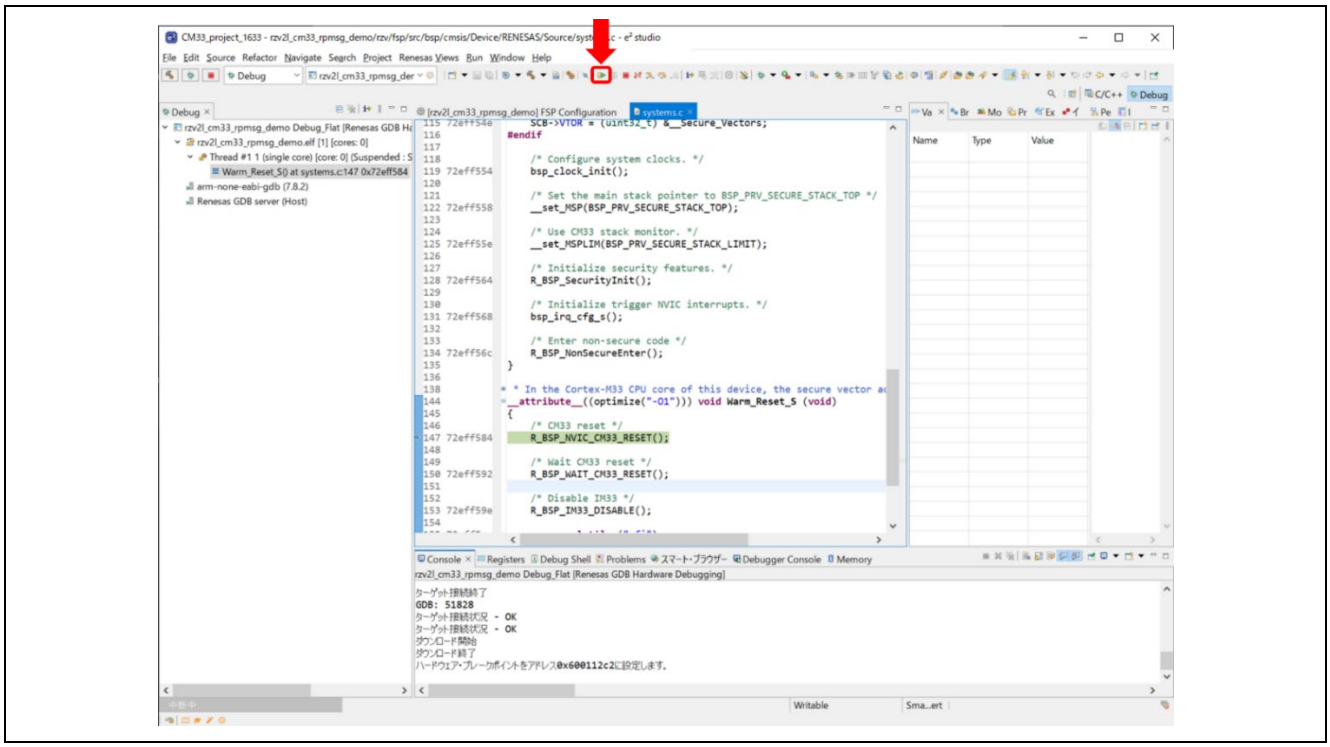


Figure 4-7. How to start to debug sample program (1)

4. The program should stop at the top of main function. Thus, please click the same button as the previous step.

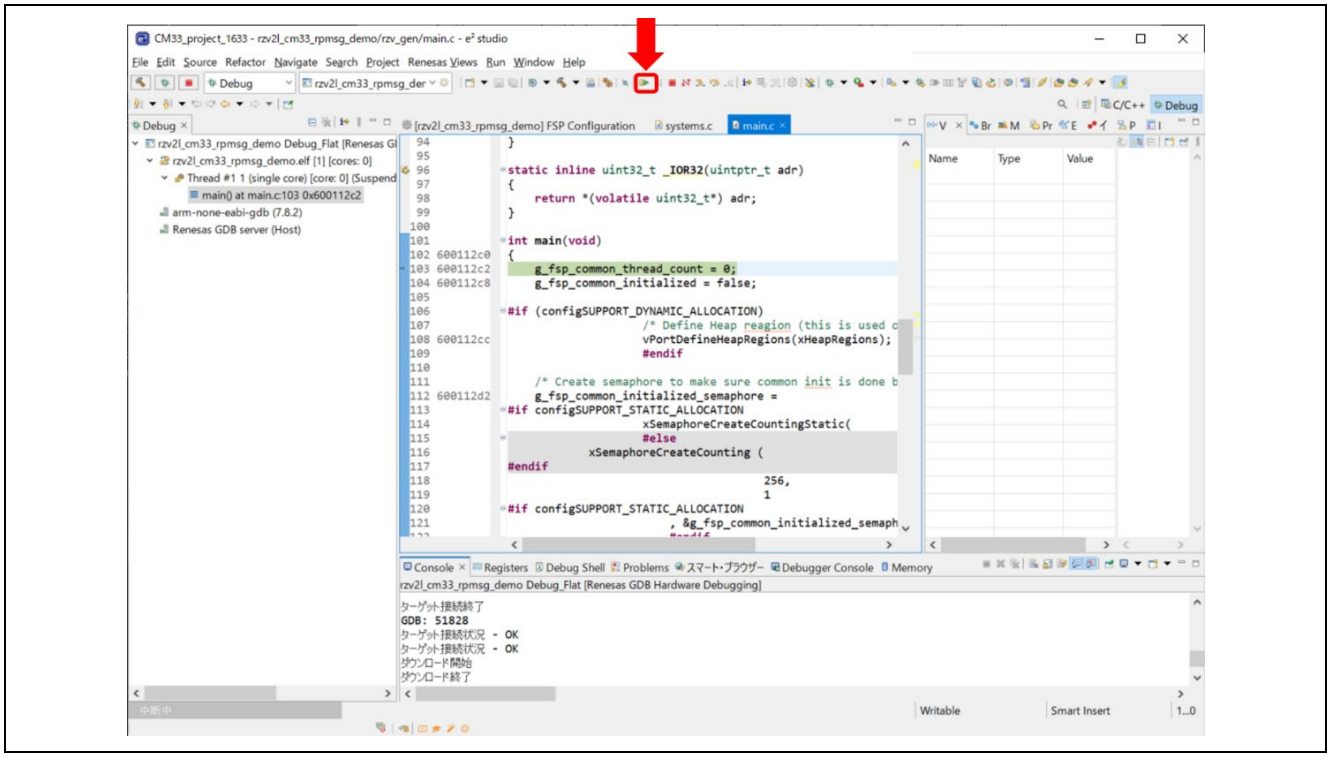


Figure 4-8. How to start to debug sample program (2)

5. Now that CM33 sample program has been started, the following message is shown on the console connected to Pmod USBUART:

```
Successfully probed IPI device
Successfully open uio device: 42F00000.rsctbl.
Successfully added memory device 42F00000.rsctbl.
Successfully open uio device: 43000000.vring-ctl0.
Successfully added memory device 43000000.vring-ctl0.
Successfully open uio device: 43200000.vring-shm0.
Successfully added memory device 43200000.vring-shm0.
Initialize remoteproc successfully.
creating remoteproc virtio
initializing rpmsg vdev
```

CM33 program is waiting for the establishment of rpmsg channel between CM33 and CA55.

4.3.2 CM33 Sample Program Invocation with u-boot

You can invoke CM33 sample program from u-boot by following the procedure described below:

1. Copy the binary files generated at 8 of section 4.2 to microSD card.
2. Insert the microSD card into CN10 of SMARC carrier board.
3. Turn on SMARC EVK by pressing reset button (i.e., SW14)
4. You should now see the following message in the console connected to CN14 of SMARC carrier board:

```
U-Boot 2021.10 (Dec 15 2023 - 06:47:44 +0000)

CPU:   Renesas Electronics CPU rev 1.0
Model: smarc-rzv2l
DRAM:  1.9 GiB
WDT:   watchdog@0000000012800800
WDT:   Started with servicing (60s timeout)
MMC:   sd@11c00000: 0, sd@11c10000: 1
Loading Environment from MMC... OK
In:    serial@1004b800
Out:   serial@1004b800
Err:   serial@1004b800
U-boot WDT started!
Net:   eth0: ethernet@11c20000
Hit any key to stop autoboot: 2
=>
```

Then, you need to hit any key to stop autoboot within 3 sec.

5. Load the binary files listed in 1. from microSD card to RAM by executing the commands below on the console. Here, N denotes the partition number in which you stored those binaries.

```
dcache off
mmc dev 1
fatload mmc 1:N 0x0001FF80 rzv2l_cm33_rpmsg_demo_secure_vector.bin
fatload mmc 1:N 0x42EFF440 rzv2l_cm33_rpmsg_demo_secure_code.bin
fatload mmc 1:N 0x00010000 rzv2l_cm33_rpmsg_demo_non_secure_vector.bin
fatload mmc 1:N 0x40010000 rzv2l_cm33_rpmsg_demo_non_secure_code.bin
cm33 start_debug 0x1001FF80 0x00010000
dcache on
```

6. Now that CM33 program has been started to run. With respect to the behavior of sample program, please see 4.5.

4.4 CA55 Sample Program Invocation

You need to follow the procedure shown below to invoke CA55 sample program running on Linux.

1. Boot up Linux by executing the following command on u-boot:

```
run bootcmd
```

2. Login as “root”

```
smarc-rzv2l login: root
```

3. Run CA55 sample program by executing the following command on Linux.

```
root@smarc-rzv2l:~# rpmsg_sample_client
```

4. Then, you can see the following message on the console relative to CN14 of SMARC carrier board. Be sure that you invoke CM33 program in advance.

```
Successfully probed IPI device
metal: info:      metal_uio_dev_open: No IRQ for device 42f00000.rsctbl.
Successfully open uio device: 42f00000.rsctbl.
Successfully added memory device 42f00000.rsctbl.
metal: info:      metal_uio_dev_open: No IRQ for device 43000000.vring-ctl0.
Successfully open uio device: 43000000.vring-ctl0.
Successfully added memory device 43000000.vring-ctl0.
metal: info:      metal_uio_dev_open: No IRQ for device 43200000.vring-shm0.
Successfully open uio device: 43200000.vring-shm0.
Successfully added memory device 43200000.vring-shm0.
metal: info:      metal_uio_dev_open: No IRQ for device 43100000.vring-ctl1.
Successfully open uio device: 43100000.vring-ctl1.
Successfully added memory device 43100000.vring-ctl1.
metal: info:      metal_uio_dev_open: No IRQ for device 43500000.vring-shm1.
Successfully open uio device: 43500000.vring-shm1.
Successfully added memory device 43500000.vring-shm1.
metal: info:      metal_uio_dev_open: No IRQ for device 42f01000.mhu-shm.
Successfully open uio device: 42f01000.mhu-shm.
Successfully added memory device 42f01000.mhu-shm.
Initialize remoteproc successfully.
Initialize remoteproc successfully.

**************************************
*   rpmsg communication sample program   *
**************************************

1. communicate with RZ/V2L CM33 ch0
2. communicate with RZ/V2L CM33 ch1

e. exit

please input
>
```

5. Type 1 if RPMsg channel 0 is used on CM33 RPMsg sample program under default setting. Otherwise, type2 for staring the communication.

4.5 Overview of Sample Program Behavior

The behavior of sample program is as follows:

1. Wait until a communication channel between CA55 and CM33 is established.
2. Once the communication channel is established, CA55 sample program starts to send the message to CM33 with incrementing its size from the minimum value 17 to the maximum value 488. At that time, the message like the following should be shown in the console connected to CN14 of SMARC carrier board:

```
Sending payload number 148 of size 165
```

3. When CM33 receives the message sent from CA55, the echo reply is sent back to CA55.
4. When CA55 receives the echo reply, the message below should be displayed in the console connected to CN14 of SMARC carrier board:

```
echo test: sent : 165
 received payload number 148 of size 165
```

5. After the message which has 488 bytes sized payload is sent from CA55 to CM33 and CM33 sends back the echo reply, the message for terminating the communication channel is sent from CA55 to CM33. Then, CA55 and CM33 sample programs output the following log messages to the corresponding consoles respectively when receiving the termination message.

- Message shown on CA55 side:

```
****************************************
*   rpmsg communication sample program   *
****************************************

1. communicate with RZ/V2L CM33 ch0
2. communicate with RZ/V2L CM33 ch1

e. exit

please input
>
```

If you would like to quit the application, please type e.

- Termination message on CM33 side

```
De-initializating remoteproc
```

6. Then, CM33 side re-waits for the establishment of connection channel. You can see the following log on the console a short time later:

```
creating remoteproc virtio
initializing rpmsg vdev
```

5. Sample Program Invocation on RZ/V2H EVK

5.1 Hardware setup

Connect J-Link to RZ/V2H EVK. For details, please refer to Getting Started with RZ/V Flexible Software Package.

5.2 CM33/CR8 Sample Program Setup

Here are the procedures for setting up the sample program running on CM33 and CR8:

1. Extract **r01an7254ej0210-rzv-multi-os-pkg.zip** on your development PC.
2. Extract either of **rzv2h_cm33_rpmsg_demo.zip** or **rzv2h_cr8_ rpmsg_demo.zip** included in **r01an7254ej0210-rzv-multi-os-pkg.zip**.
3. Open e^2 studio 2024-07 and click File > Import.
4. Double-click General and select Existing Projects into Workspace as shown in Figure 5-1:

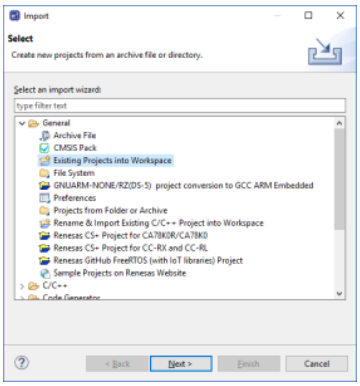


Figure 5-1. Import sample project (RZ/V2H) (1)

5. Input the path to the directory of the sample program you would like to import, press Enter key and click Finish button.

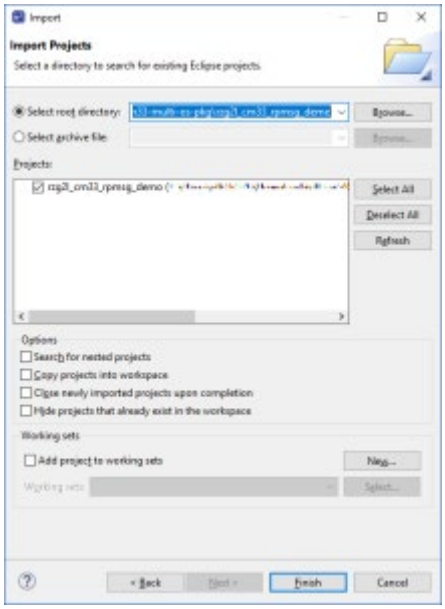


Figure 5-2. Import sample project (RZ/V2H) (2)

6. (Optional) By default RPMsg channel 0 and 1 are configured to be used on CM33 and CR8 respectively. If you would like to change the channel, open the property of MainTask#0 on FSP Smart Configurator, you need to specify the number of channel you would like to use for Thread Context and to push **Generate Project Content** button. If **Generate Project Content** pop-up is shown, click **Proceed** to reflect the change to the source code.

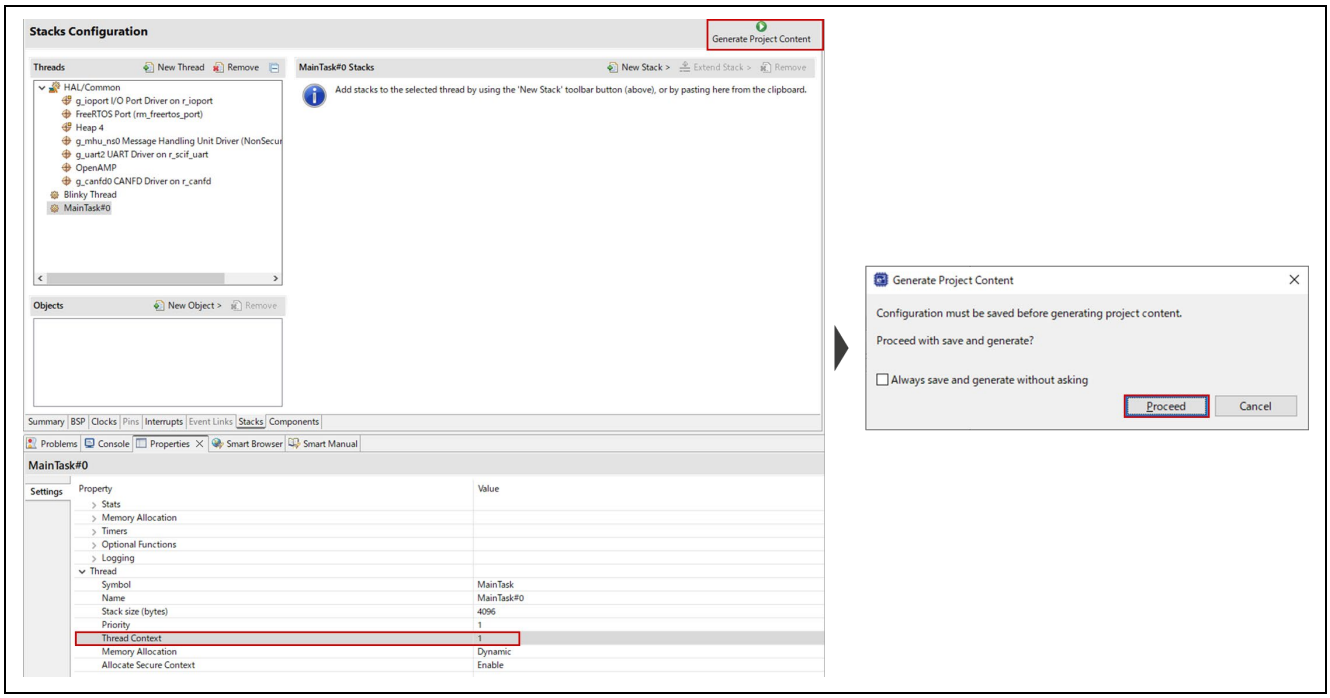


Figure 5-3. RPMsg Channel Setting (RZ/V2H)

7. Build the project from Choose Project > Build Project.
8. If the build is successfully completed, build artifacts as listed below should be generated in Debug and/or Release directory of the project you imported in accordance with the active Build Configuration.
   - rzv2h_<cpu>_rpmsg_demo.elf
   - rzv2h_<cpu>_rpmsg_demo.bin

   <cpu> stands for either cm33 or cr8.

5.3 Note for CR8 Sample Program

- CR8 Sample Program expects that CR8 and CM33 Sample Program Projects are imported to the same e^2 studio workspace. Otherwise, FSP Smart Configurator won't work for CR8 Sample Program.
- Please note that the CR8 Sample program is created for CR8 Core0, not for CR8 Core1.

5.4 CM33/CR8 Sample Program for the communication with Linux

5.4.1 CM33/CR8 Sample Program Invocation using J-Link

Carry out the procedures as shown below for setting up CM33 and/or CR8 sample program for communicating with Linux running on CA55:

1. Click **Debug** button shown below:

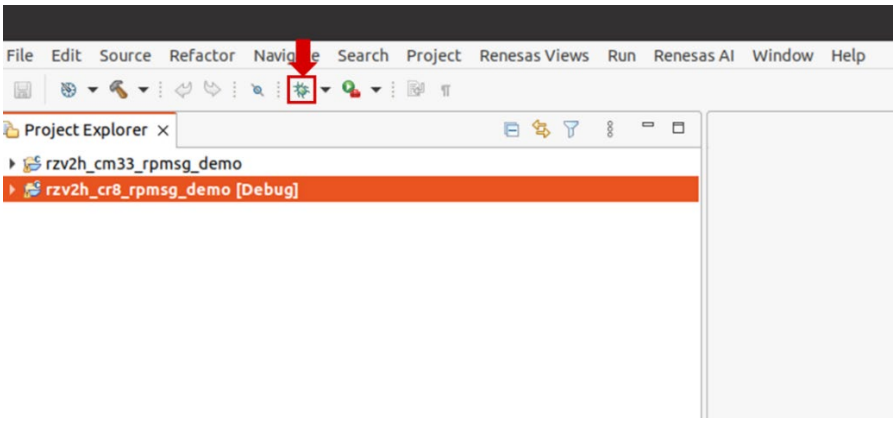


Figure 5-4. Select of Debug Configuration (RZ/V2H)

2. For the first time, you may need to choose rzv2h_<cpu>_rpmsg_demo Debug_Flat or rzv2h_<cpu>_rpmsg_demo Release_Flat in accordance with the build configuration you are using. At that time, choose the appropriate debug configuration and click OK.

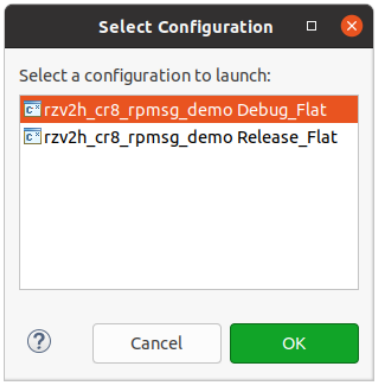


Figure 5-5. Debug Function Launch

If Confirmation Perspective Switch window below appears, please press Switch to go ahead.

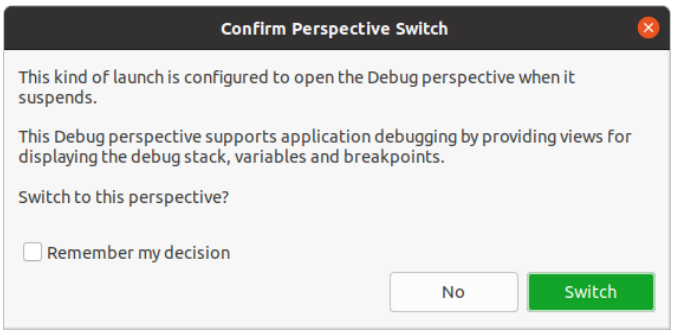


Figure 5-6. Confirmation window to open the Debug Perspective

3. When the debug perspective is opened, Program Counter (PC) should be located at the top of Warm_Reset_S and Reset Handler for CM33 and CR8 e^2 studio project respectively. Then, you need to press the button in as shown in Figure 5-7.

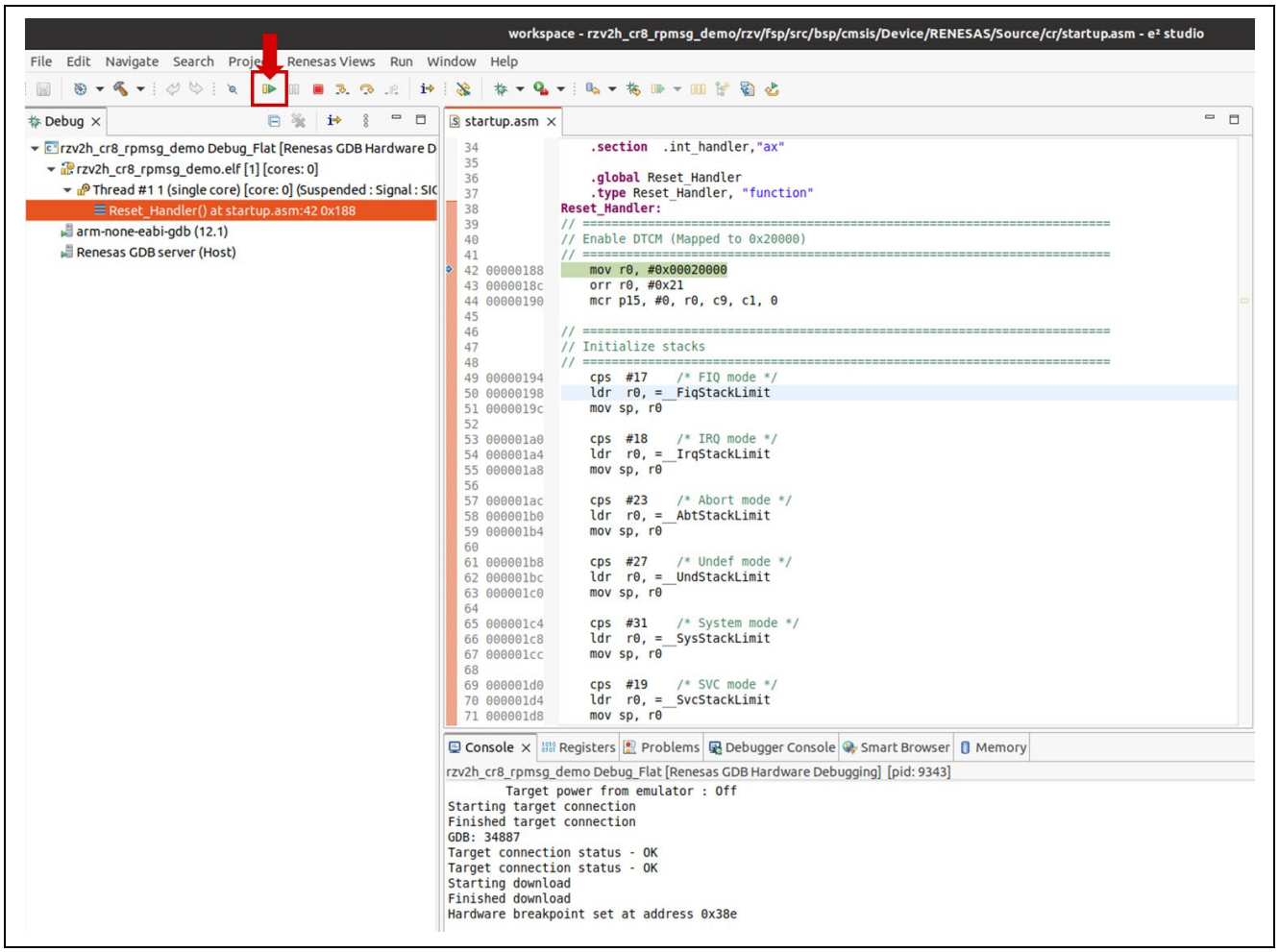


Figure 5-7. How to start to debug Sample Program (1)

4. The program should stop at the top of main function. Then, click the same button to continue.

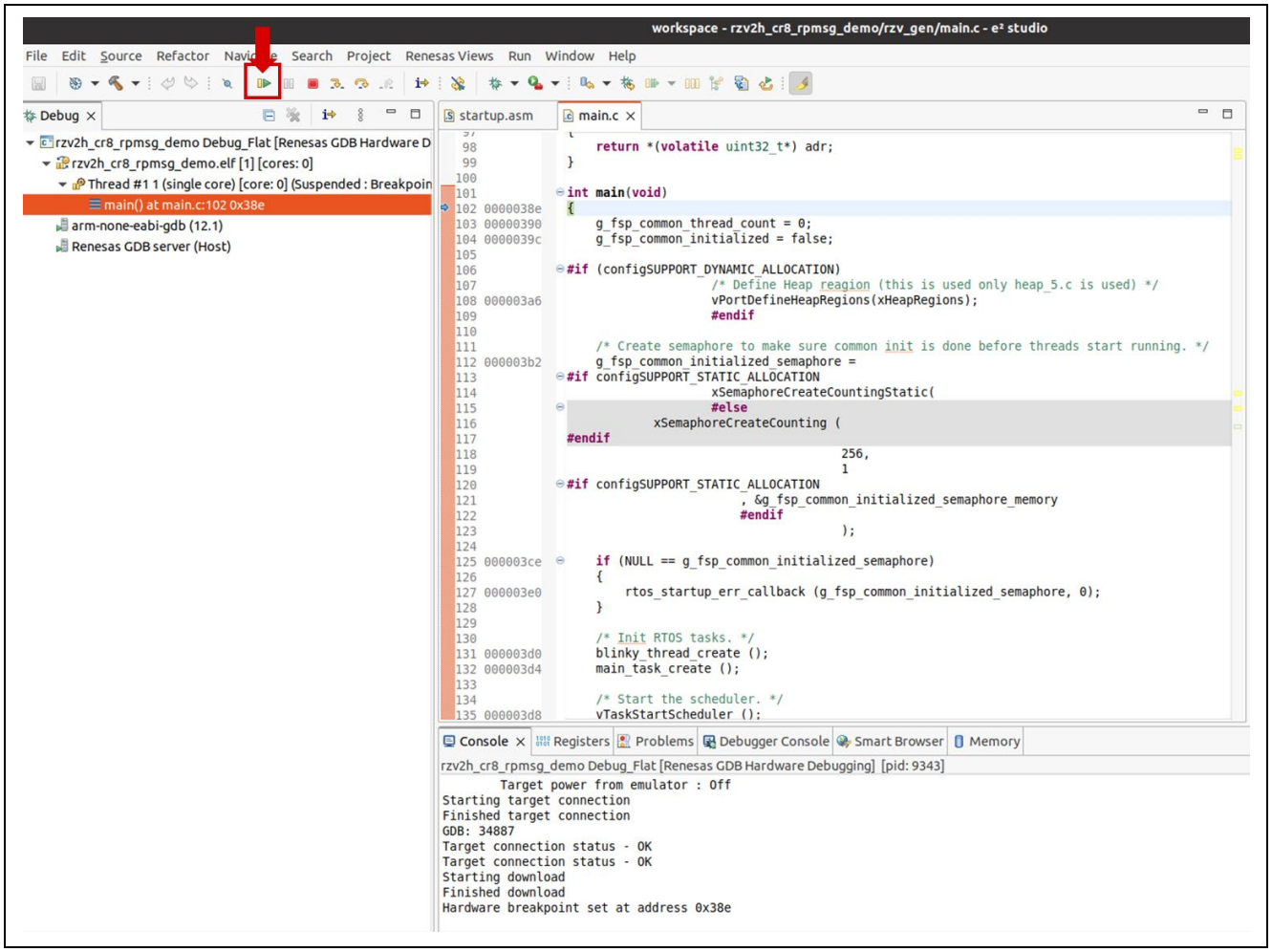


Figure 5.8. How to start to debug Sample Program (2)

5. CM33 and/or CR8 sample program start and waiting for the establishment of RPMsg channel with CA55.

5.4.2 CM33/CR8 Sample Program Invocation from u-boot

On RZ/V2H EVK, you can invoke RPMsg Sample Program from u-boot. Here is the example of procedure:

1. Place rzv2h_<cpu>_rpmsg_demo.bin in SD card where Linux kernel and Device Tree Blob are stored.
2. Insert SD card into SD1 of RZ/V2H EVK.
3. Turn on RZ/V2H EVK. Then, you should see the following message in the console connected to CN12 of RZ/V2H EVK.

```
U-Boot 2021.10 (Jun 14 2024 - 18:14:19 +0000)

CPU:   Renesas Electronics CPU rev 1.0
Model: Renesas EVK Version 1 based on r9a09g057h4
DRAM:  15.9 GiB
MMC:   mmc@15c00000: 0, mmc@15c10000: 1

(snip)

Net:   eth0: ethernet@15c30000, eth1: ethernet@15c40000
Hit any key to stop autoboot:  3
```

4. Hit any key within 3 sec. to stop autoboot.
5. Carry out the following setup on u-boot to kick CM33 and/or CR8.

- For CM33

```
=> setenv cm33start 'dcache off; mw.l 0x10420D2C 0x02000000; mw.l 0x1043080c
0x08003000; mw.l 0x10430810 0x18003000; mw.l 0x10420604 0x00040004; mw.l
0x10420C1C 0x00003100; mw.l 0x10420C0C 0x00000001; mw.l 0x10420904 0x00380008;
mw.l 0x10420904 0x00380038; ext4load mmc 0:2 0x08001e00
boot/rzv2h_cm33_rpmsg_demo.bin; mw.l 0x10420C0C 0x00000000; dcache on'
=> saveenv
=> run cm33start
```

- For CR8

```
=> setenv cr8start 'dcache off; mw.l 0x10420D24 0x04000000; mw.l 0x10420600
0xE000E000; mw.l 0x10420604 0x00030003; mw.l 0x10420908 0x1FFF0000; mw.l
0x10420C44 0x003F0000; mw.l 0x10420C14 0x00000000; mw.l 0x10420908 0x10001000;
mw.l 0x10420C48 0x00000020; mw.l 0x10420908 0x1FFF1FFF; mw.l 0x10420C48
0x00000000; ext4load mmc 0:2 0x12040000 boot/rzv2h_cr8_rpmsg_demo_itcm.bin;
ext4load mmc 0:2 0x08180000 boot/rzv2h_cr8_rpmsg_demo_sram.bin; ext4load mmc
0:2 0x40800000 boot/rzv2h_cr8_rpmsg_demo_sdram.bin; mw.l 0x10420C14
0x00000003; dcache on;'
=> saveenv
=> run cr8start
```

5.5 CA55 Sample Program Invocation

1. Boot up Linux by executing the command stated below on u-boot for example:

```
=> run bootcmd
```

2. Login as “root” once Linux is booted up:

```
rzv2h-evk-ver1 login: root
```

3. Run RPMsg Sample Program by the command below:

```
root@rzv2h-evk-ver1:~# rpmsg_sample_client
```

4. When Sample Program started to work successfully, you can see the following log on your console:

```
[XXX] proc_id:0 rsc_id:0 mbx_id:1
metal: info:      metal_uio_dev_open: No IRQ for device 10480000.mbox-uio.
[XXX] Successfully probed IPI device
metal: info:      metal_uio_dev_open: No IRQ for device 42f00000.rsctbl.
[XXX] Successfully open uio device: 42f00000.rsctbl.
[XXX] Successfully added memory device 42f00000.rsctbl.
metal: info:      metal_uio_dev_open: No IRQ for device 43000000.vring-ctl0.
[XXX] Successfully open uio device: 43000000.vring-ctl0.
[XXX] Successfully added memory device 43000000.vring-ctl0.
metal: info:      metal_uio_dev_open: No IRQ for device 43200000.vring-shm0.
[XXX] Successfully open uio device: 43200000.vring-shm0.
[XXX] Successfully added memory device 43200000.vring-shm0.
metal: info:      metal_uio_dev_open: No IRQ for device 43100000.vring-ctl1.
[XXX] Successfully open uio device: 43100000.vring-ctl1.
[XXX] Successfully added memory device 43100000.vring-ctl1.
metal: info:      metal_uio_dev_open: No IRQ for device 43500000.vring-shm1.
[XXX] Successfully open uio device: 43500000.vring-shm1.
[XXX] Successfully added memory device 43500000.vring-shm1.
metal: info:      metal_uio_dev_open: No IRQ for device 42f01000.mhu-shm.
[XXX] Successfully open uio device: 42f01000.mhu-shm.
[XXX] Successfully added memory device 42f01000.mhu-shm.
[XXX] Initialize remoteproc successfully.
[XXX] proc_id:1 rsc_id:1 mbx_id:1
[XXX] Initialize remoteproc successfully.
[XXX] proc_id:0 rsc_id:0 mbx_id:2
[XXX] Initialize remoteproc successfully.
[XXX] proc_id:1 rsc_id:1 mbx_id:2
[XXX] Initialize remoteproc successfully.
[XXX] proc_id:0 rsc_id:0 mbx_id:3
[XXX] Initialize remoteproc successfully.
[XXX] proc_id:1 rsc_id:1 mbx_id:3
[XXX] Initialize remoteproc successfully.

****************************************
*   rpmsg communication sample program   *
****************************************

1. communicate with CM33      ch0
2. communicate with CM33      ch1
3. communicate with CR8 core0 ch0
4. communicate with CR8 core0 ch1
5. communicate with CR8 core1 ch0
6. communicate with CR8 core1 ch1
7. communicate with CM33 ch0 and CR8 core0 ch1
8. communicate with CM33 ch0 and CR8 core1 ch1
9. communicate with CR8 core0 ch0 and CR8 core1 ch1


e. exit

please input
>
```

5. By inputting the number which performs the communication you would like to try, Sample Program starts to work. Please note that menu 5, 6, 8 and 9 is for future extension and shouldn't work on this version.

6. By typing **e**, Sample Program should be terminated with the message shown below:

```
please input
> e
[XXX] 42f00000.rsctbl closed
[XXX] 43000000.vring-ctl0 closed
[XXX] 43200000.vring-shm0 closed
[XXX] 43100000.vring-ctl1 closed
[XXX] 43500000.vring-shm1 closed
[XXX] 42f01000.mhu-shm closed
```

5.6 Overview of Sample Program Behavior

1. Wait until the valid communication channel among CA55, CM33 and/or CR8 is established.
2. Once the channel is established, CA55 RPMsg Sample Program starts to send the message to CM33 and/or CR8 with incrementing the message size from the minimum value 17 to the maximum value 488. During the communication, the message as shown below should be displayed on your console:

```
[XXX] Sending payload number 148 of size 165
```

3. When CM33 and/or CR8 receives the message sent from CA55, the echo reply is sent back to CA55.
4. When CA55 receives the echo reply, the message below should be displayed on your console:

```
[XXX] received payload number 148 of size 165
```

5. After the message which has 488 bytes sized payload is sent from CA55 to CM33/CR8 and CM33/CR8 sends back the echo reply, the message for terminating the communication channel is sent from CA55 to CM33/CR8. Then, Sample Program outputs the following log messages to your console:

```
[XXX] **********************************
[XXX] Test Results: Error count = 0
[XXX] **********************************
[XXX] Quitting application .. Echo test end
[XXX] Stopping application...
```

5.7 Note for Sample Program

When trying **7. communication with CM33 ch0 and CR8 core 0 ch1**, be sure to invoke CM33 RPMsg Sample Program first followed by CR8 Sample Program beforehand. If Sample Program works successfully, the communications between CA55 and CM33 and between CA55 and CR8 work concurrently as shown in the following log:

```
[XXX] thread start
[CR8 ] creating remoteproc virtio
[XXX] thread start
[CR8 ] initializing rpmsg shared buffer pool
[CR8 ] initializing rpmsg vdev
[CR8 ]  1 - Send data to remote core, retrieve the echo and validate its
integrity ..
[CM33] creating remoteproc virtio
[CR8 ] Remote proc init.
[XXX] cond signal 1 sync:0
[CM33] initializing rpmsg shared buffer pool
[CM33] initializing rpmsg vdev
[CM33]  1 - Send data to remote core, retrieve the echo and validate its
integrity ..
[CM33] Remote proc init.
[CR8 ] RPMSG endpoint has created. rp_ept:0xffffa4c67870
[CR8 ] register sig:2 succeeded.
[CR8 ] register sig:15 succeeded.
[CM33] RPMSG endpoint has created. rp_ept:0xffffa5468870
[CR8 ] RPMSG service has created.
[CR8 ] sending payload number 0 of size 17
[XXX] cond signal 0 sync:0
[CM33] RPMSG service has created.
[CM33] sending payload number 0 of size 17
[XXX] cond signal 1 sync:0
[CR8 ]  received payload number 0 of size 17
[XXX] cond signal 0 sync:0
[CM33]  received payload number 0 of size 17

(snip)

[CR8 ]  received payload number 469 of size 486
[CR8 ] sending payload number 470 of size 487
[331] cond signal 1 sync:0
[CR8 ]  received payload number 470 of size 487
[CR8 ] sending payload number 471 of size 488
[XXX] cond signal 1 sync:0
[CR8 ]  received payload number 471 of size 488
[CR8 ] **********************************
[CR8 ]  Test Results: Error count = 0
[CR8 ] **********************************
[XXX] cond signal 1 sync:0
[CM33] Quitting application .. Echo test end
[CM33] Stopping application...
[CR8 ] Quitting application .. Echo test end
[CR8 ] Stopping application...
```

6. Reference Documents

- R01AN6240: RZ/V2L Getting Started with Flexible Software Package
- R01US0565: RZ/V Verified Linux Package Version 3.0.5 Release Note
- R01US0617: SMARC EVK of RZ/V2L Linux Start-up Guide
- R12UZ0147: RZ/V2H Evaluation Board Kit (Secure type) Hardware Manual

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Jan.14.22	-	First edition issued.
1.01	Jun.13.22	-	Updated in align with RZ/V2L Linux Package Version 1.0.1.
1.02	Jul.31.22	-	Updated in align with RZ/V2L Verified Linux Package Version 3.0.0
1.10	Jan.31.23	-	Updated in align with RZ/V2L Verified Linux Package Version 3.0.2
1.11	Jul.24.23	-	Updated in align with RZ/V2L Verified Linux Package Version 3.0.4
1.12	Jan.31.24	-	Updated in align with RZ/V2L Verified Linux Package Version 3.0.5
2.00	May.31.24	-	Updated in align with RZ/V AI SDK v3.00 and RZ/V Verified Linux Package Version 3.0.5.
2.10	Aug.30.24	-	Updated in align with RZ/V AI SDK v5.00 and RZ/V Verified Linux Package Version 3.0.6.

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 be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

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

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

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

Trademarks

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

Contact information

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