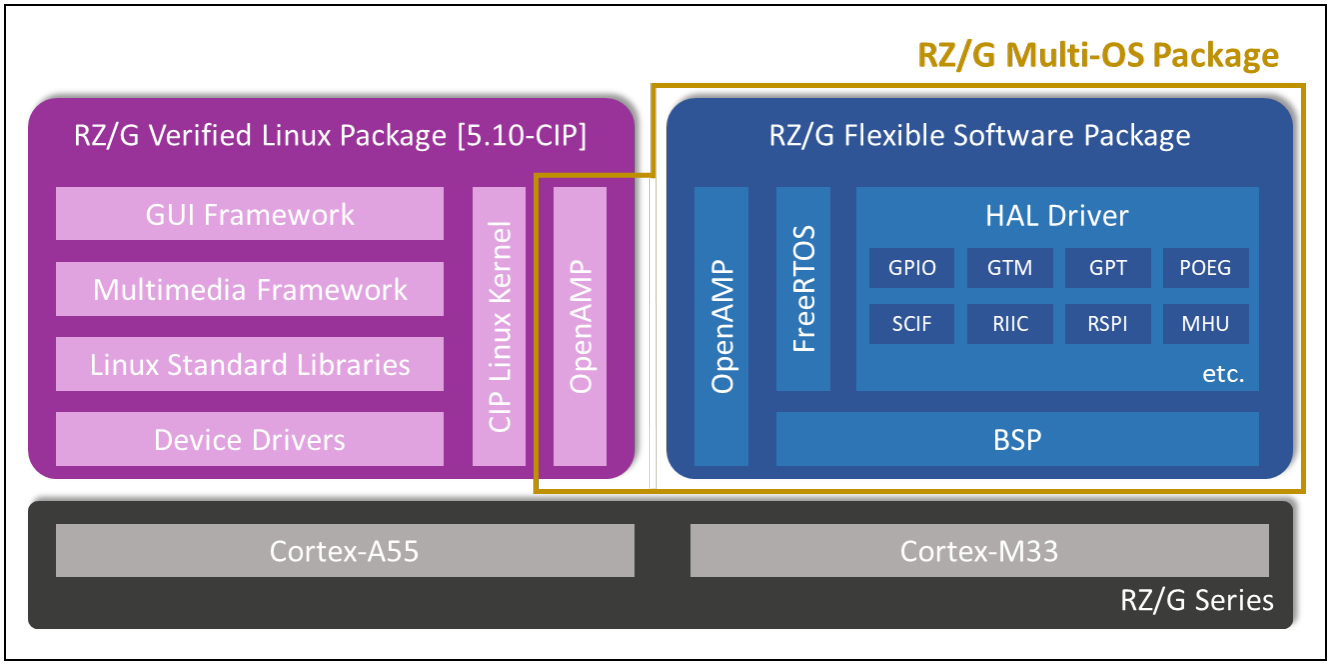
RZ/G2L, G2LC, G2UL, G3S

Release Note for RZ/G Multi-OS Package V2.3.0

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) environment and sample program showing how to implement Inter-Processor Communication between those CPU cores.

This package consists of RZ/G Flexible Software Package (hereinafter referred to as RZ/G FSP) and Inter-Processor Communication Feature Package for RZ/G Verified Linux Package (hereinafter referred to as RZ/G VLP) or RZ/G3S Linux Board Support Package (hereinafter referred to as RZ/G3S Linux BSP).



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

- **RZ/G FSP supporting RZ/G2L, RZ/G2LC, RZ/G2UL and RZ/G3S**
  The software package consists of production ready peripheral drivers, FreeRTOS and portable middleware stacks and the best in-case HAL drivers with low memory footprint.
- **OpenAMP**
  The framework includes the software components needed for Asymmetric Multiprocessing (AMP) systems such as Inter-Processor Communication.

Target Device

RZ/G2L, RZ/G2LC, RZ/G2UL and RZ/G3S

Contents

1. Specifications

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

Table 1-1. List of used Peripheral Modules

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 for RZ/G2L, G2LC, G2UL and G3S

Item	Contents
Microprocessor used	RZ/G2L, RZ/G2LC, RZ/G2UL and RZ/G3S
Integrated Development Environment	e^2 studio 2024-10
C compiler	GNU Arm Embedded 10.3 2021.10

3. Multi-OS Package Setup

3.1 RZ/G Flexible Software Package Setup

Please refer to Getting Started with Flexible Software Package.

3.2 OpenAMP related stuff Integration for RZ/G2L, RZ/G2LC and RZ/G2UL

This section describes how to integrate OpenAMP related stuff to RZ/G Verified Linux Package 3.0.6 (hereinafter referred to as “VLP/G”) for RZ/G2L, RZ/G2LC and RZ/G2UL. For details on how to build the VLP/G for RZ/G2L, RZ/G2LC and RZ/G2UL, please refer to SMARC EVK of RZ/G2L, RZ/G2LC, RZ/G2UL Linux Start-up Guide incorporated in it.

1. Follow the procedure from the beginning of **2.2 Building Images** to **(4) Add layers** of **SMARC EVK of RZ/G2L, RZ/G2LC and RZ/G2UL Linux Start-up Guide**.
2. Download Multi-OS Package (r01an5869ej0230-rzg-multi-os-pkg.zip) to a working directory and run the commands stated below:

```
$ cd ~/rzg_vlp_<pkg ver> (Note 1)
$ unzip <Multi-OS Dir>/r01an5869ej0230-rzg-multi-os-pkg.zip (Note 2)
$ tar zxvf r01an5869ej0230-rzg-multi-os-pkg/meta-rz-features_multi-os_v2.3.0.tar.gz
```

Notes: 1. <pkg ver> stands for the version number of VLP/G (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-rzg2l
```

(Optional for remoteproc support)

4. Apply the modification stated below to **meta-rz/-features/meta-rz-multi-os/meta-rzg2l/recipes-kernel/linux-renesas_5.10.bbappend** for enabling remoteproc support.

```
FILESEXTRAPATHS_prepend := "${THISDIR}/${PN}:"

ENABLE_REMOTEPROC = "1"
```

5. Start a build as described in **(6) Start a build** of **2.2 Building Images** in **SMARC EVK of RZ/G2L, RZ/G2LC and RZ/G2UL Linux Start-up Guide**.

```
$ MACHINE=<board> bitbake core-image-<target>
```

Here, allowable values for <board> are as stated below:

- smarc-rzg2l for RZ/G2L Evaluation Board Kit PMIC version
- smarc-rzg2lc for RZ/G2LC Evaluation Board Kit
- smarc-rzg2ul for RZ/G2UL Evaluation Board Kit

Also, **minimal**, **bsp**, **weston** or **qt** can be specified for <target>.
For details, please refer to **SMARC EVK of RZ/G2L, RZ/G2LC and RZ/G2UL Linux Start-up Guide**.

3.3 OpenAMP related stuff Integration for RZ/G3S

This section describes how to integrate OpenAMP related stuff to VLP/G for RZ/G3S.

1. Follow the procedure from the beginning of **2.2 Building Images** to **(3) Add layers** of **SMARC EVK of RZ/G3S Linux Start-up Guide**.

(Optional for CM33 cold boot support)

2. Apply the patch for CM33 cold boot support.

```
$ cd ~/rzg_vlp_<pkg ver>
$ cp <patch placed folder>/0001-bl2-cm33-coldboot-support.patch .
$ patch -p1 < ./0001-bl2-cm33-coldboot-support.patch
```

3. Download Multi-OS Feature Package (r01an5869ej0230-rzg-multi-os-pkg.zip) to your working directory and run the commands stated below:

```
$ cd ~/rzg_vlp_<pkg ver>
$ unzip <Multi-OS download dir>/r01an5869ej0230-rzg-multi-os-pkg.zip
$ tar zxvf r01an5869ej0230-rzg-multi-os-pkg/meta-rz-features_multi-os_v2.3.0.tar.gz
```

4. Add the layer for Multi-OS Package.

```
$ bitbake-layers add-layer ../meta-rz-features/meta-rz-multi-os/meta-rzg3s
```

(Optional for remoteproc support)

5. Apply the modification stated below to **meta-rz/-features/meta-rz-multi-os/meta-rzg3s/recipes-kernel/linux-renesas_5.10.bbappend** for enabling remoteproc support.

```
FILESEXTRAPATHS_prepend := "${THISDIR}/${PN}:"

ENABLE_REMOTEPROC = "1"
```

6. Start a build as described in **(5) Start a build** of **2.2 Building Images** in **SMARC EVK of RZ/G3S Linux Start-up Guide**.

```
$ MACHINE=smarc-rzg3s bitbake core-image-<target>
```

Here, **minimal** or **bsp** can be specified for <target>. For details, please refer to **SMARC EVK of RZ/G3S Linux Start-up Guide**.

3.4 Note for integration to RZ/G VLP

On RZ/G VLP, the peripherals which is NOT used enters Module Standby Mode. That means the peripherals used on CM33 side might NOT become worked implicitly. To avoid this situation, we prepare for the patch below for RZ/G2Lx, RZ/G2UL and RZ/G3S, respectively:

- 0006-clk-renesas-r9a07g044-Set-SCIF1-SCIF2-OSTM2.patch for RZ/G2L and RZ/G2LC
- 0008-clk-renesas-r9a07g043-Set-OSTM2.patch for RZ/G2UL
- 0001-Set-SCIF1-and-OSTM1-OSTM2-as-critical-clock.patch for RZ/G3S

For example, 0006-clk-renesas-r9a07g044-Set-SCIF1-SCIF2-OSTM2.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 following modification in red to the patches.

```
drivers/clk/renesas/r9a07g044-cpg.c | 3 ++
 1 file changed, 3 insertions(+)

diff --git a/drivers/clk/renesas/r9a07g044-cpg.c
b/drivers/clk/renesas/r9a07g044-cpg.c
index e27caa075af7a..f3cda2e05757f 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_SCIF1_CLK_PCK,
+        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:

- https://github.com/renesas-rz/rz_linux-cip/blob/rz-5.10-cip41/drivers/clk/renesas/r9a07g044-cpg.c#L221-L398

3.5 Deployment of RZ/G VLP

3.5.1 In case of configuring CA55 as Boot CPU

Please follow **3. Preparing the SD Card**, **4. Reference Board Setting** and **5. Booting and Running Linux** of **SMARC EVK of RZ/G2L, RZ/G2LC and RZ/G2UL Linux Start-up Guide** and **SMARC EVK of RZ/G3S Linux Start-up Guide** for deploying bootloader files, Linux kernel image, device tree and rootfs.

3.5.2 In case of configuring CM33 as Boot CPU on RZ/G3S

If CM33 is configured as Boot CPU, you need to follow the procedure stated below for deploying bootloader files, Linux kernel image, device tree and rootfs:

1. Follow **3. Preparing the SD Card**, **4.1 Preparation of Hardware and Software**, **4.2 Startup Procedure** and **4.3 Download Flash Writer to RAM** of **SMARC EVK of RZ/G3S Linux Start-up Guide** to Invoke Flash Writer.

(Optional for Flash Writer settings)

2. Change the transfer rate of Flash Writer from the default one (115200bps) to the high speed one (921600bps) as shown below:

```
> sup
Scif speed UP
Please change to 921.6Kbps baud rate setting of the terminal.
```

3. Program **bl2_no_bp_spi-smarc-rzg3s.srec** with Flash Writer as shown below:

```
> XLS2
===== Qspi writing of RZ/G3 Board Command =============
Load Program to Spiflash
Writes to any of SPI address.
Program size & Qspi Save Address
===== Please Input Program Top Address ============
Please Input : H'a1e00
===== Please Input Qspi Save Address ===
Please Input : H'200000
Please send ! ('.' & CR stop load)
```

Send **bl2_no_bp_spi-smarc-rzg3s.srec** via terminal software (e.g., TeraTerm) after the message **Please send ! ('.' & CR stop load)** is output on your console as shown above.

When **bl2_no_bp_spi-smarc-rzg3s.srec** is programmed successfully, the following message is shown on your console:

```
Erase SPI Flash memory...
Erase Completed
Write to SPI Flash memory.
======= Qspi Save Information =================
SpiFlashMemory Stat Address : H'00200000
SpiFlashMemory End Address  : H'0021EB58
================================================

SPI Data Clear(H'FF) Check : H'00000000-0000FFFF,Clear OK?(y/n)
```

Finally, type **y** to continue.

4. Program **fip-smarc-rzg3s.srec** with Flash Writer as shown below:

```
> XLS2
===== Qspi writing of RZ/G3 Board Command =============
Load Program to Spiflash
Writes to any of SPI address.
Program size & Qspi Save Address
===== Please Input Program Top Address ============
Please Input : H'0
===== Please Input Qspi Save Address ===
Please Input : H'264000
Please send ! ('.' & CR stop load)
```

Send **fip-smarc-rzg3s.srec** via terminal software (e.g., TeraTerm) after the message **Please send ! ('.' & CR stop load)** is output on your console as shown above.

When **fip-smarc-rzg3s.srec** is programmed successfully, the following message is shown on your console:

```
Erase SPI Flash memory...
Erase Completed
Write to SPI Flash memory.
======= Qspi Save Information =================
SpiFlashMemory Stat Address : H'00264000
SpiFlashMemory End Address  : H'00342B6E
================================================

SPI Data Clear(H'FF) Check : H'00000000-0000FFFF,Clear OK?(y/n)
```

Finally, type **y** to continue.

4. Sample Program Invocation on RZ/G2L, RZ/G2LC and RZ/G2UL SMARC EVK

4.1 RZ/G2L, RZ/G2LC and RZ/G2UL Hardware Setup

1. Connect J-Link to RZ/G2L SMARC EVK. For details, please refer to Getting Started with 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. (Note)

Note: There is no valid SCIF channel connected to Pmod 0 or Pmod 1 on RZ/G2UL Smarc EVK.

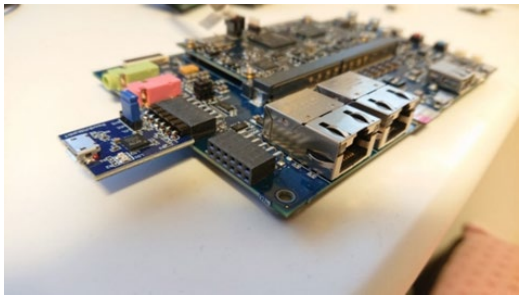


Figure 4-1. Connection between RZ/G2L SMARC EVK and Pmod USBUART

4.2 RZ/G3S Hardware Setup

1. Connect J-Link to RZ/G3S SMARC EVK. For details, please refer to Getting Started with Flexible Software Package.
2. Connect Pmod USBUART to the upper side of PMOD 3A connector of Smarc EVK as shown below for securing the console for sample program.

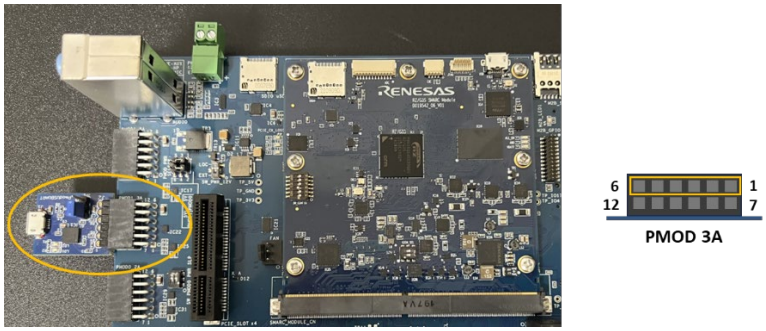


Figure 4-2. Connection between Pmod USBUART and RZ/G3S SMARC EVK

4.3 CM33 Sample Program Setup

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

1. Extract r01an5869ej230-rzg-multi-os-pkg.zip on your development PC.
2. Extract <device>_rpmsg_<com_type>_demo.zip there. Here, <device> should be any of **rzg2l_cm33**, **rzg2lc_cm33**, rzg2ul_cm33, **rzg3s_cm33** or **rzg3s_cm33_fpu**. Also, <com_type> should either of **linux-rtos** or **rtos-rtos**.
3. Open e^2 studio 2024-10 and click **File** > **Import**.
4. Double-click General and select Existing Projects into Workspace as shown in Figure 4-3:

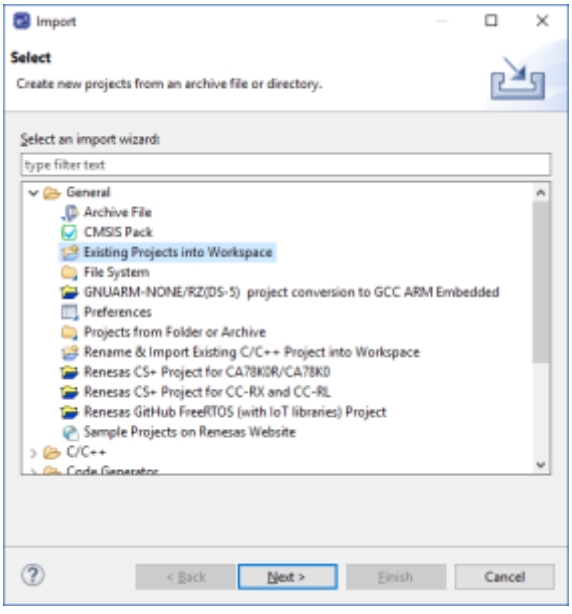


Figure 4-3. Import sample project (1)

5. Input the path to the directory where RPMsg sample program is put, press Enter key and click Finish button.

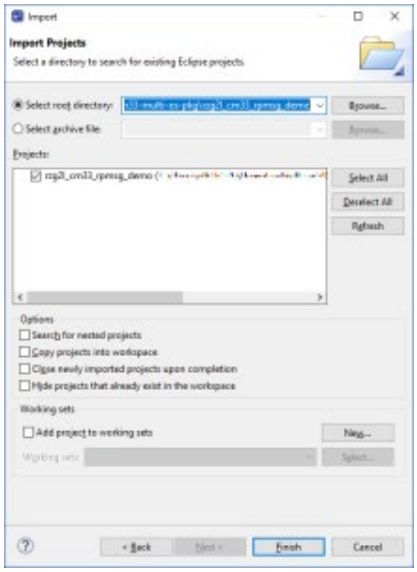


Figure 4-4. Import sample project (2)

(Optional for remoteproc support)

6. Change the value for ENABLE_REMOTEPROC defined in platform_info.h of <device>_rpmsg_<com _type>_demo.zip from 0 to 1.

```
#define ENABLE_REMOTEPROC (1U)
```

(Optional for CM33 cold boot support)

7. Extract cm33coldboot_patch.zip and overwrite **src** directory in sample program with the extracted one.
8. Add QSPI (r_xspi_qspi) driver at Stacks tab.

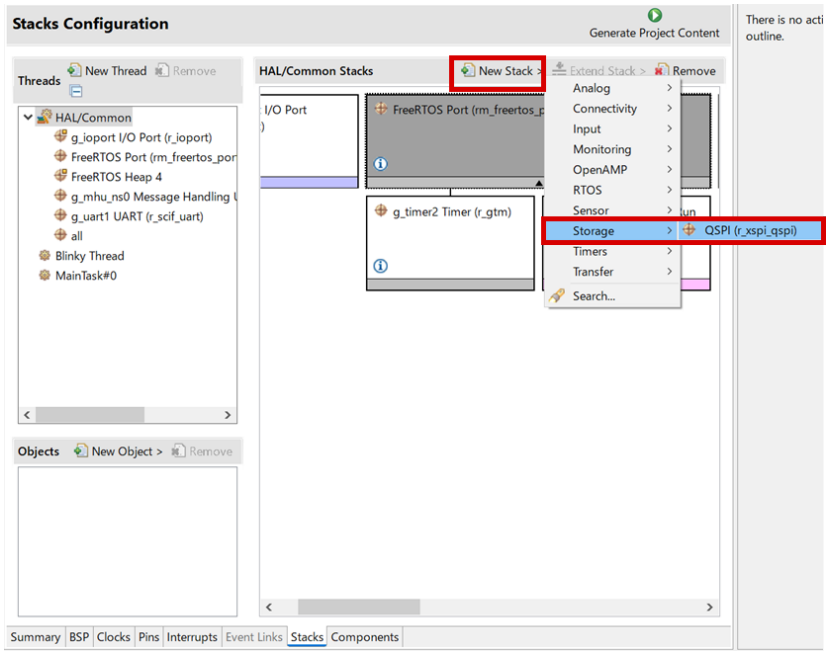


Figure 4-5. Add QSPI (r_xspi_qspi) Driver

(Optional for CM33 cold boot support)

9. Choose QSPI driver you added in the previous step and click Properties as shown below:

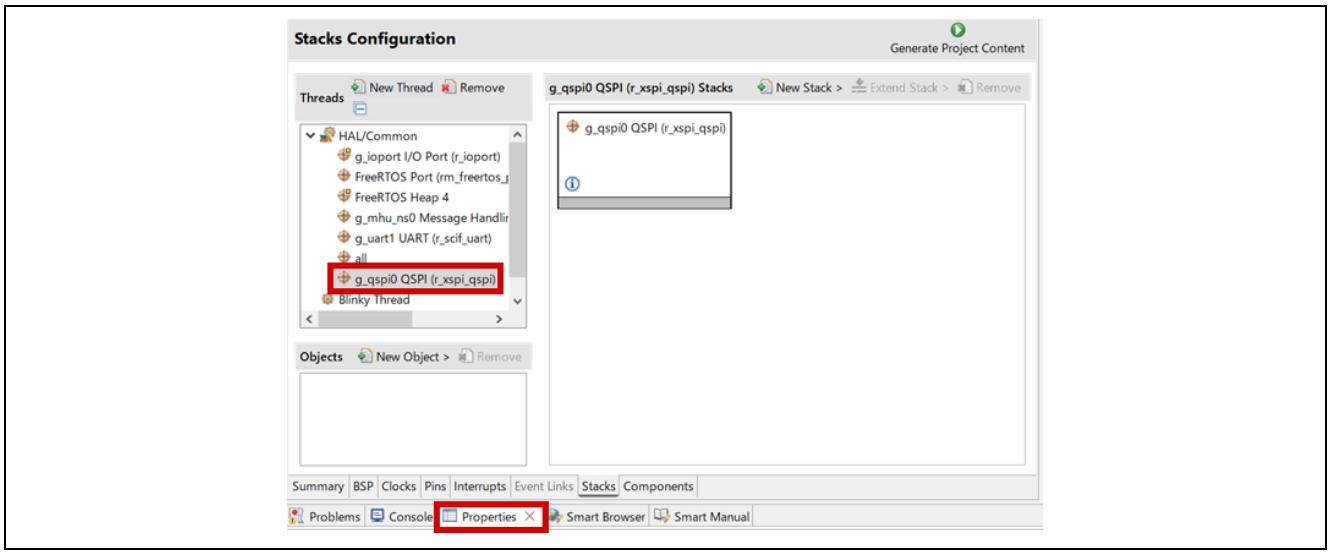


Figure 4-6. QSPI Driver Property Setting (1)

(Optional for CM33 cold boot support)

10. Configure **Prefetch Function**, **XIP Enter M7-M0**, **XIP Exit M7-M0**, **Pre initialize function** and **Post initialize function** as **Enable**, **0xAA**, **0x55**, **xspi_pre_init** and **xspi_post_init**, respectively.

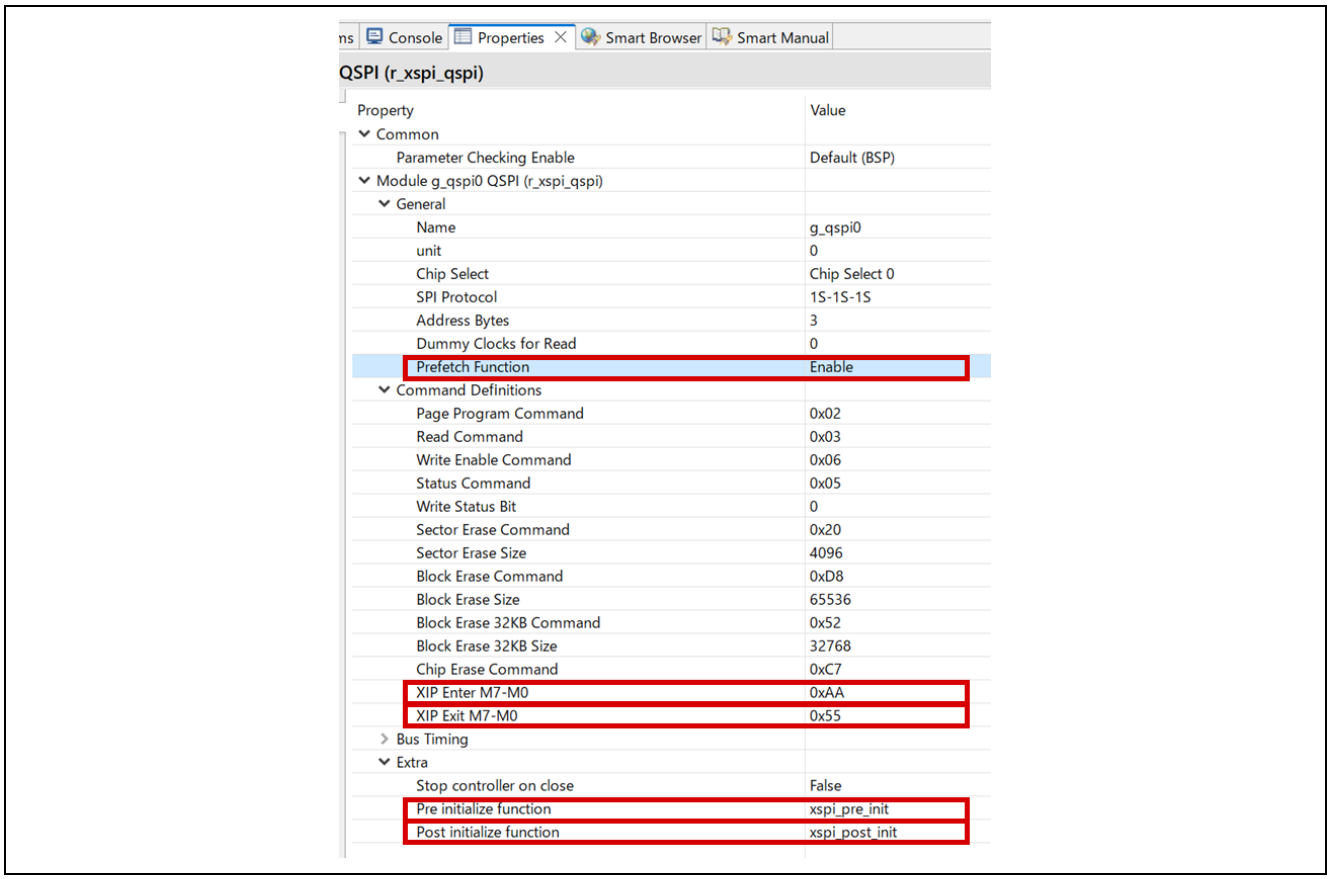


Figure 4-7. QSPI Driver Property Setting (2)

(Optional for CM33 cold boot support)

11. Click **Generate Project Content** to generate the updated source code:

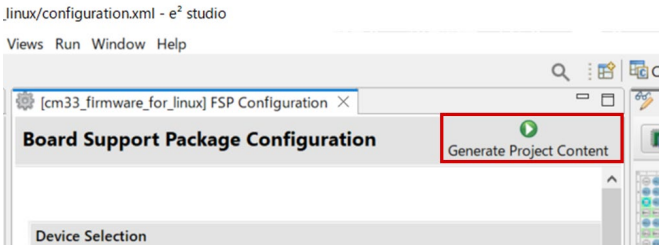


Figure 4-8. Generate Project Content

12. Build the project from Choose **Project** > **Build Project**.
13. 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.
    - <device>_rpmsg_linux-rtos_demo.elf (Note1)
    - <device>_rpmsg_linux-rtos_demo_non_secure_code.bin (Note 2)
    - <device>_rpmsg_linux-rtos_demo_non_secure_vector.bin (Note 2)
    - <device>_rpmsg_linux-rtos_demo_secure_code.bin (Note 2)
    - <device>_rpmsg_linux-rtos_demo_secure_vector.bin (Note 2)
    - <device>_rpmsg_linux-rtos_demo.srec (Note 3)
    - <device>_rpmsg_rtos-rtos_demo.elf (Note 3)
    - <device>_rpmsg_rtos-rtos_demo.srec (Note 3)

Notes: 1. The possible string for <device> is any of rzg2l_cm33, rzg2lc_cm33, rzg2ul_cm33, rzg3s_cm33 and rzg3s_cm33_fpu.
2. The possible string is any of rzg2l_cm33, rzg2lc_cm33 and rzg2ul_cm33.
3. The possible string is either rzg3s_cm33 or rzg3s_cm33_fpu.

4.4 CM33 Sample Program Invocation

4.4.1 CM33 Sample Program Invocation with Segger J-Link

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

(Optional for CM33 cold boot support)

1. Select **Run** > **Debug Configurations...** if CM33 is configured as Boot CPU for RZ/G3S.

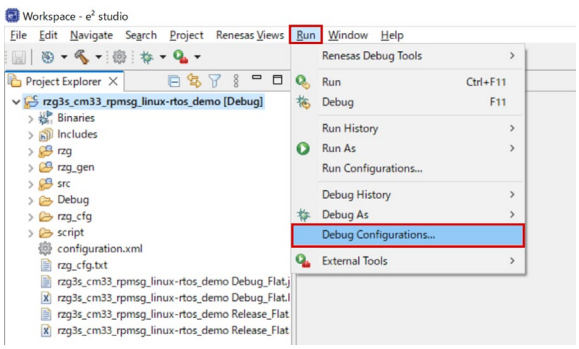


Figure 4-9. Debug Configuration (1)

(Optional for CM33 cold boot support)

2. Click **rzg3s_cm33_rpmsg_demo Debug_Flat** or **rzg3s_cm33_rpmsg_demo Release_Flat**.

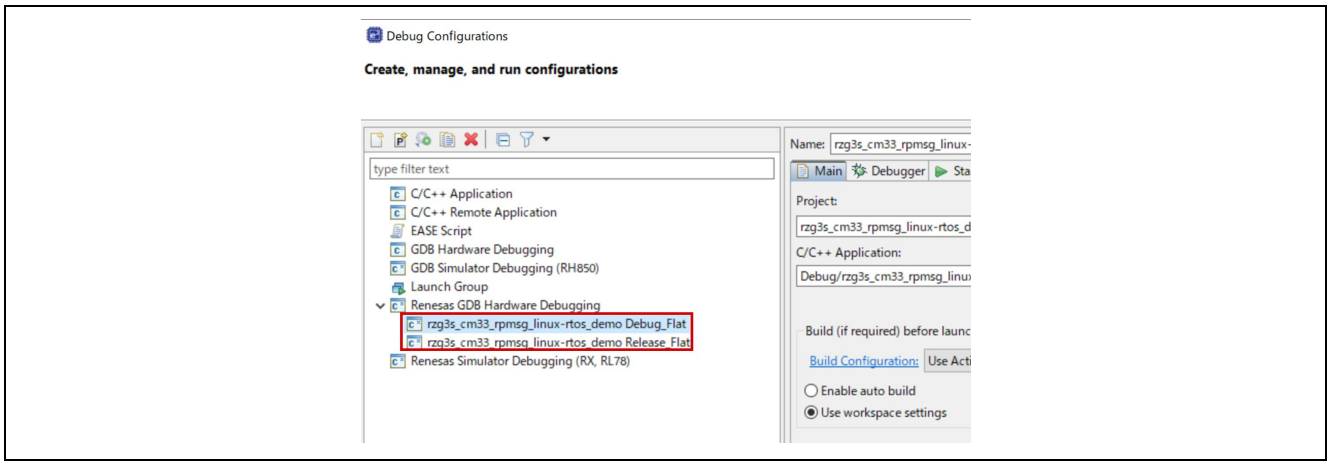


Figure 4-10. Debug Configuration (2)

(Optional for CM33 cold boot support)

3. Click **Debugger** tab and select **Connection Settings** as shown below:

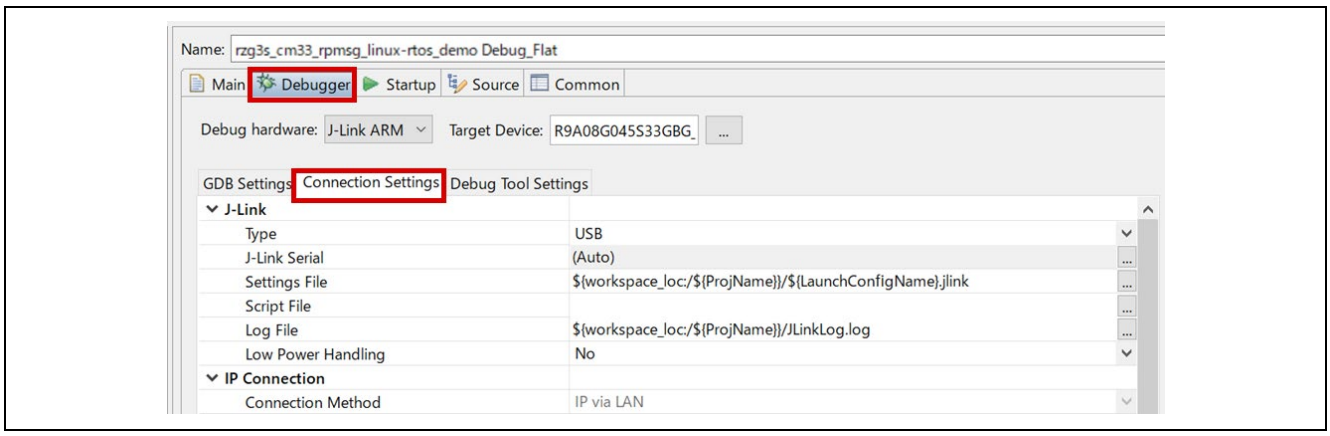


Figure 4-11. Debugger - Connection Settings

(Optional for CM33 cold boot support)

4. Configure **Reset after download** as **Yes**.

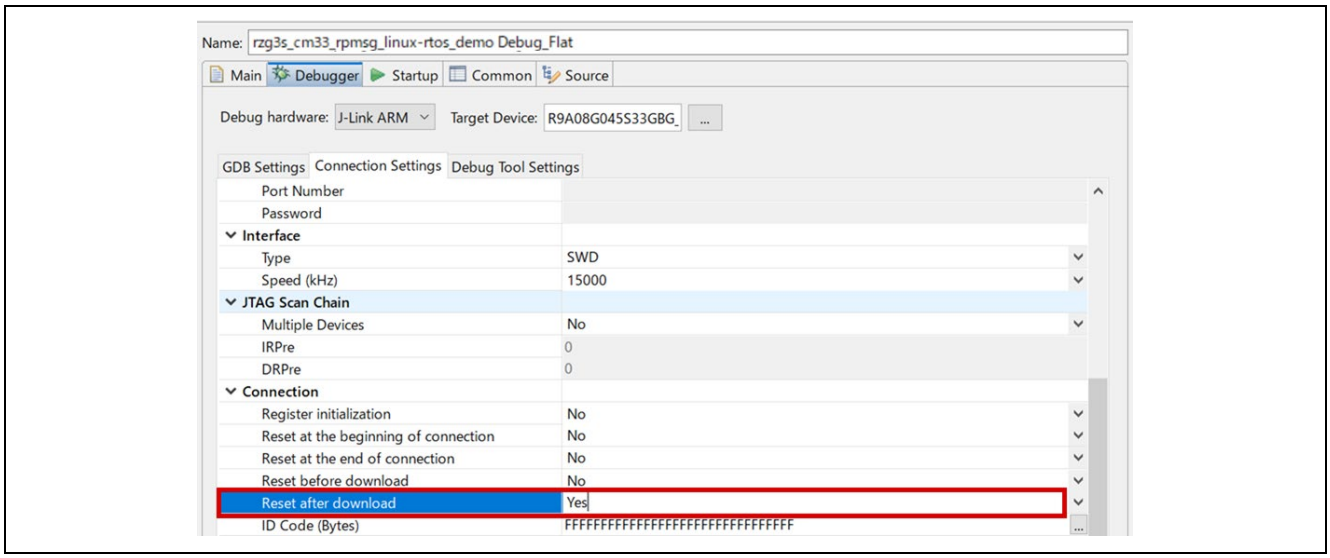


Figure 4-12. Reset after download Setting

(Optional for CM33 cold boot support)

5. Select **Startup** tab and click **Add...** as shown below:

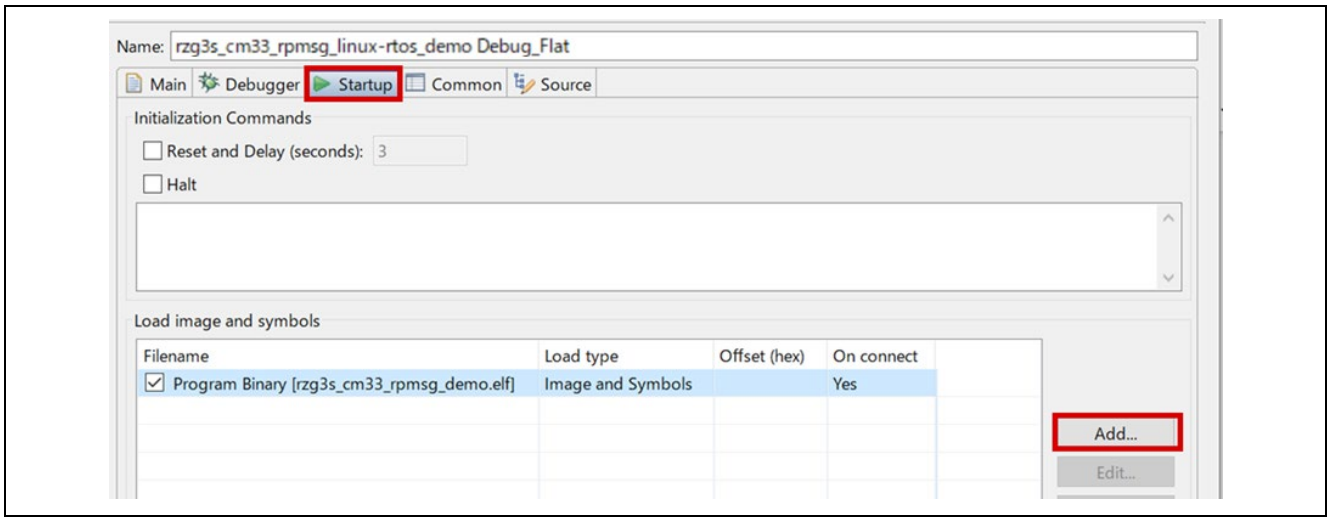


Figure 4-13. Add download module (1)

(Optional for CM33 cold boot support)

6. Click **Workspage...**, choose **rzg3s_cm33_rpmsg_demo_cm33boot.srec** and click **OK**.

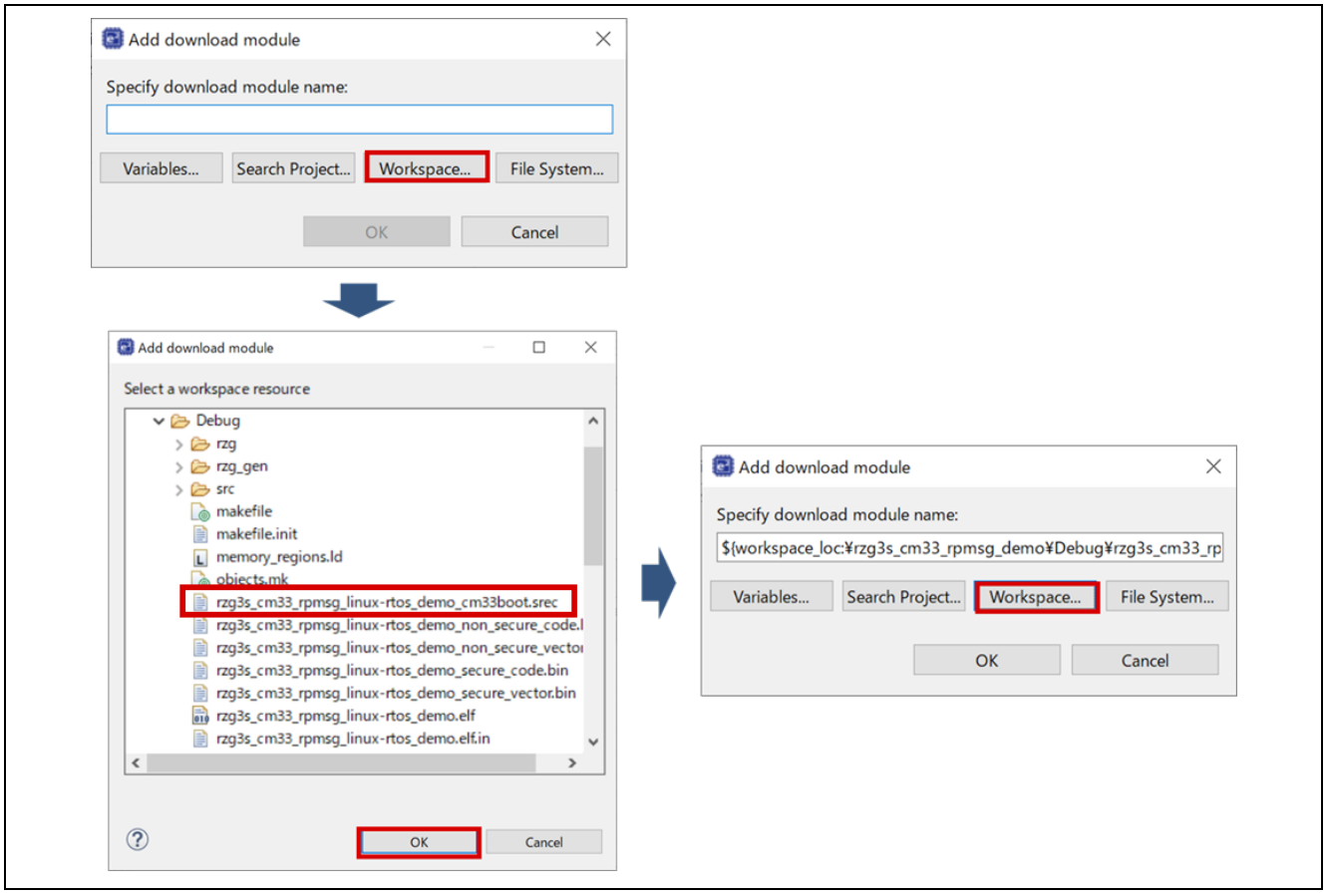


Figure 4-14. Add download module (2)

(Optional for CM33 cold boot support)

7. Configure **Load type** of **Program Binary [rzg3s_cm33_rpmsg_demo.elf]** and **rzg3s_cm33_rpmsg_demo_cm33boot.srec** as **Symbols only** and **Image only**, respectively.

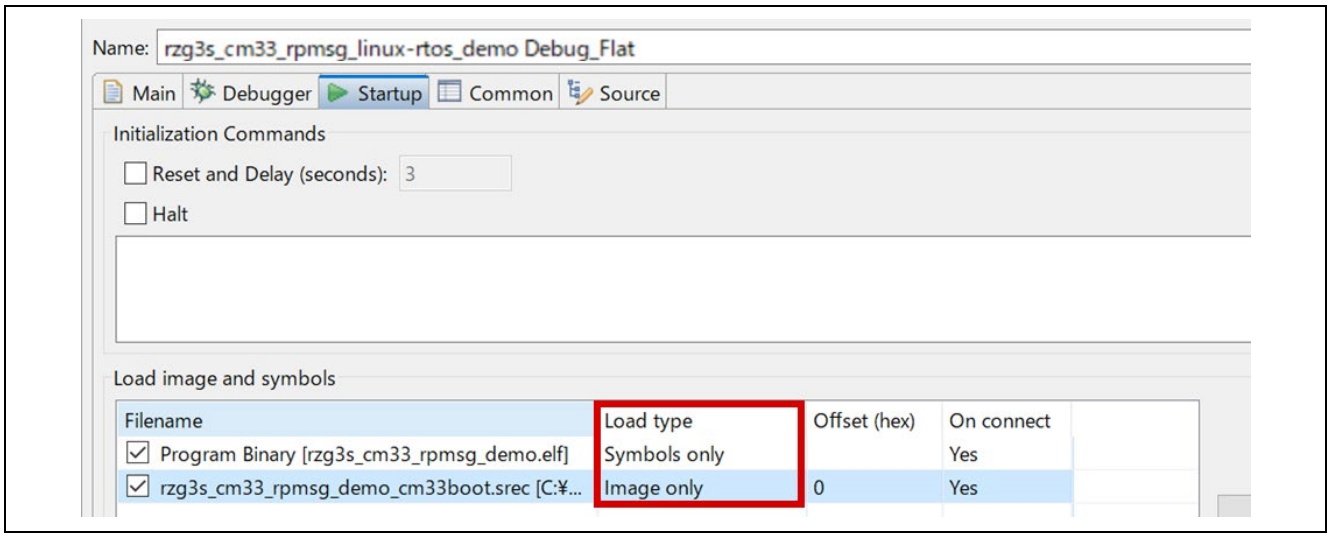


Figure 4-15. Configuration of Load type

8. Click Debug button to launch Debug Perspective.

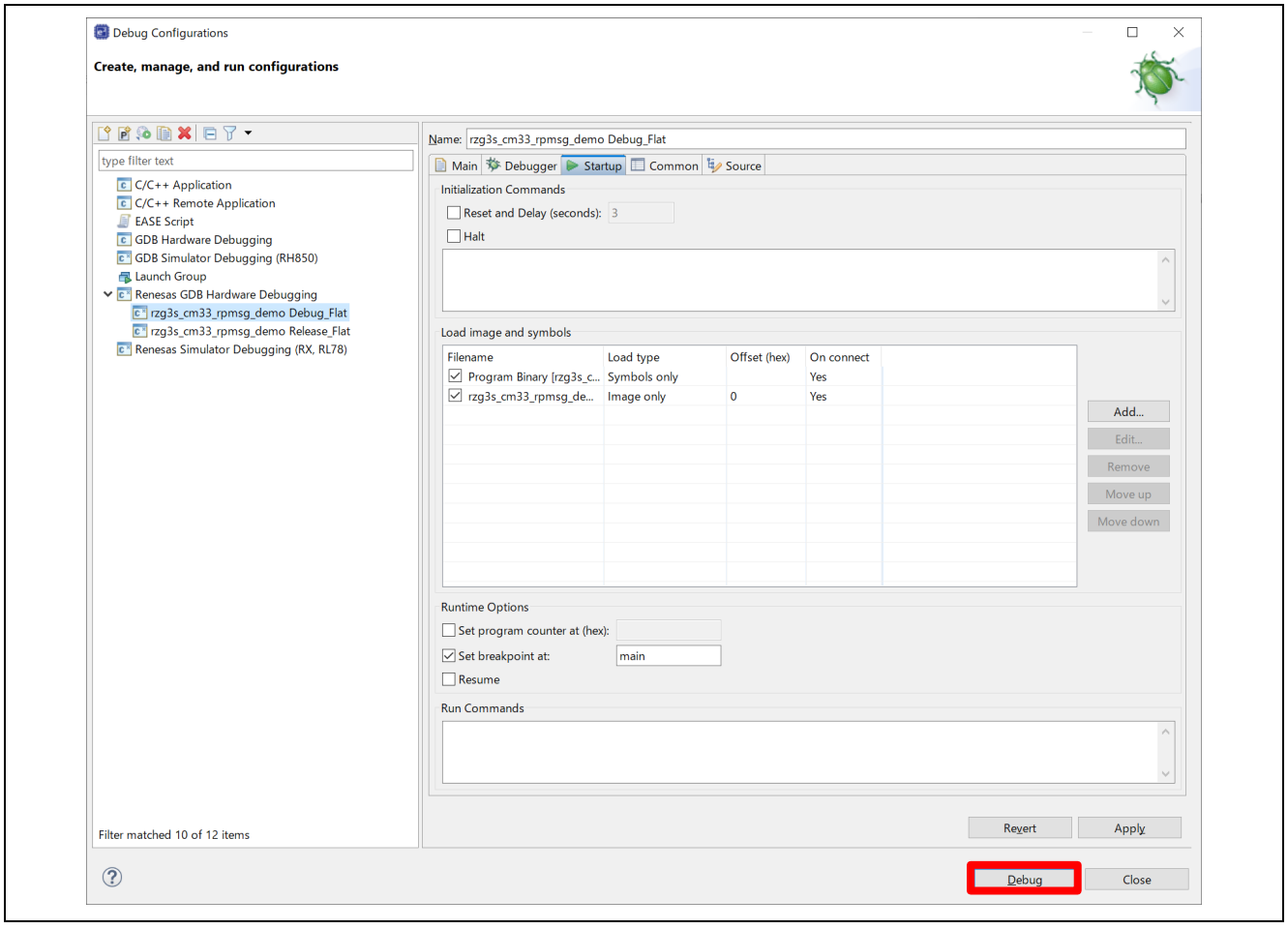


Figure 4-16. Debug Perspective Launch

If **Confirmation Perspective Switch** window below apprears, press **Switch** to continue:

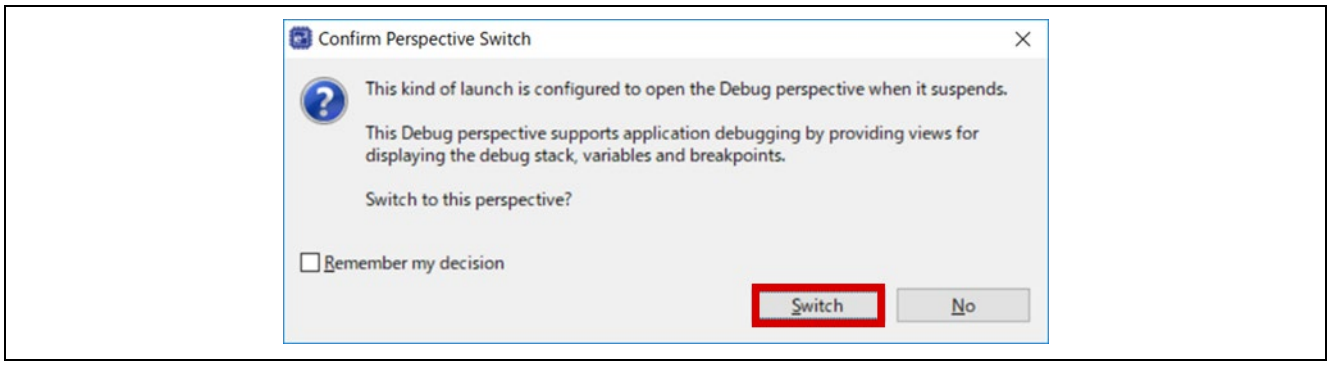


Figure 4-17. Confirm Perspective Switch

9. 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 4-17:

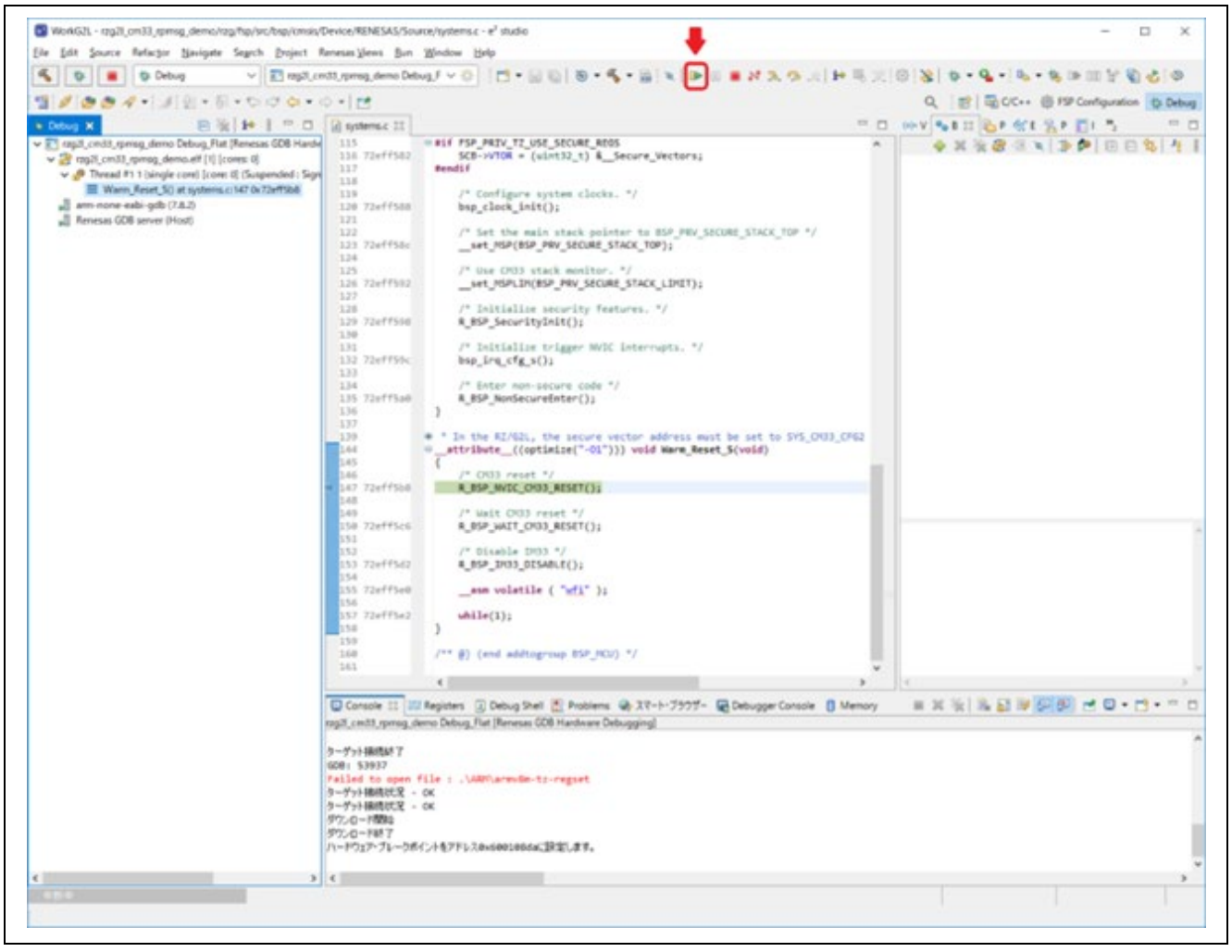


Figure 4-18. Start to debug RPMgs Sample Program (1)

10. The program should stop at the top of the **main** function. Then, click the same button as the previous step.

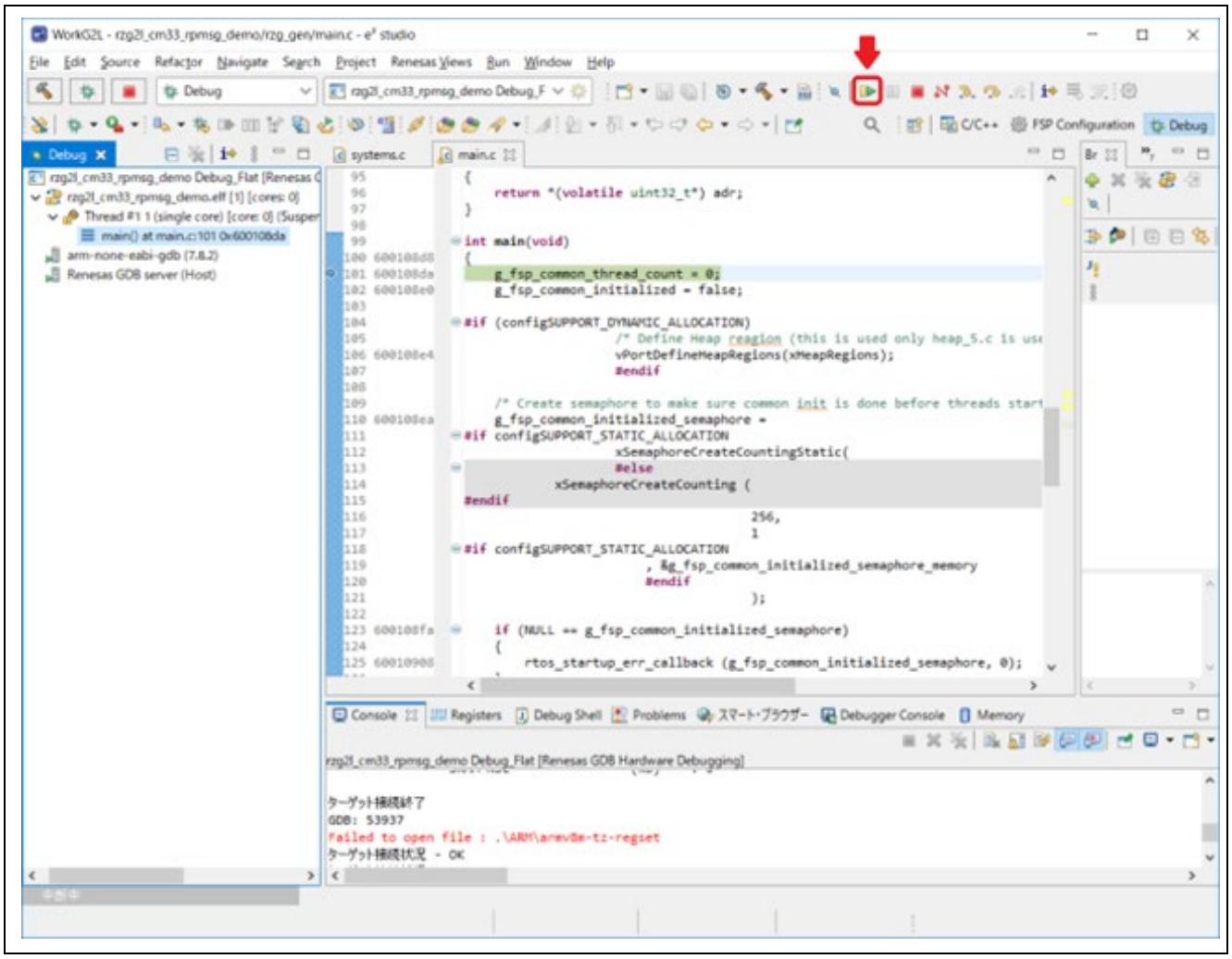


Figure 4-19. Start to debug RPMgs Sample Program (2)

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

```
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
```

At this point of time, the CM33 program is waiting for the establishment of rpmsg channel with CA55.

Note: There is no valid SCIF channel connected to Pmod 0 or Pmod 1 on RZ/G2UL SMARC EVK and so, any message won't be shown with RZ/G2UL.

4.4.2 CM33 Sample Program Invocation with u-boot

On RZ/G2L, G2LC and G2UL SMARC EVK, you can invoke RPMsg sample program from u-boot by following the procedure described below:

1. Copy <device>_rpmsg_demo_secure_code.bin, <device>_rpmsg_demo_secure_vector.bin, <device>_rpmsg_demo_non_secure_code.bin and <device>_rpmsg_demo_non_secure_vector.bin generated at 7 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 (Sep 20 2023 - 02:21:30 +0000)

CPU:   Renesas Electronics K rev 1.0
Model: smarc-rzg2l
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, 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 rzg2l_cm33_rpmsg_demo_secure_vector.bin
fatload mmc 1:N 0x42EFF440 rzg2l_cm33_rpmsg_demo_secure_code.bin
fatload mmc 1:N 0x00010000 rzg2l_cm33_rpmsg_demo_non_secure_vector.bin
fatload mmc 1:N 0x40010000 rzg2l_cm33_rpmsg_demo_non_secure_code.bin
cm33 start_debug 0x1001FF80 0x00010000
dcache on
```

6. Now that CM33 program has started to run. With respect to the behavior of the sample program, please see 4.6.

4.4.3 CM33 Sample Program Invocation with remoteproc for RZ/G2L, RZ/G2LC and RZ/G2UL

On RZ/G2L, G2LC and G2UL SMARC EVK, you can invoke RPMsg sample program with remoteproc by following the procedure stated below:

1. Booting up Linux by following **5. Booting and Running Linux** of **SMARC EVK of RZ/G2L, RZ/G2LC and RZ/G2UL Linux Start-up Guide**
2. Invoke the command below to specify RPMsg sample program to be loaded:

```
root@smarc-rzg2l:~# echo rzg2l_cm33_rpmsg_linux-rtos_demo.elf >
/sys/class/remoteproc/remoteproc0/firmware
```

3. Kick CM33 by invoking the command below:

```
root@smarc-rzg2l:~# echo start > /sys/class/remoteproc/remoteproc0/state
```

If CM33 starts to work successfully, the following message should be shown:

```
root@smarc-rzg2l:~# echo start > /sys/class/remoteproc/remoteproc0/state
[ 737.289773] remoteprocremoteproc0: powering up cm33
[ 737.348226] remoteprocremoteproc0: Booting fwimage rzg2l_cm33_rpmsg_demo.elf, size 1347660
[ 737.356732] remoteprocremoteproc0: unsupported resource 4
[ 737.366255] remoteproc0#vdev0buffer: assigned reserved memory node vdev0buffer@0x43200000
[ 737.374784] remoteproc0#vdev0buffer: registered virtio0 (type 7)
[ 737.380989] remoteprocremoteproc0: remote processor cm33 is now up
```

4.4.4 CM33 Sample Program Invocation with remoteproc for RZ/G3S

On RZ/G3S SMARC EVK, you can invoke RPMsg sample program with remoteproc by following the procedure stated below:

1. Booting up Linux by following **5. Booting and Running Linux** of **SMARC EVK of RZ/G3S Linux Start-up Guide**
2. Invoke the command below to specify RPMsg sample program to be loaded:

```
root@smarc-rzg3s:~# echo <device>_rpmsg_linux-rtos_demo.elf > /sys/class/remoteproc/remoteprocX/firmware
(Note 1)(Note 2)
```

3. Kick CM33 by invoking the command below:

```
root@smarc-rzg3s:~# echo start > /sys/class/remoteproc/remoteprocX/state
```

If CM33 starts to work successfully, the following message should be shown:

```
root@smarc-rzg3s:~# echo start > /sys/class/remoteproc/remoteproc0/state
[  152.569813] remoteproc remoteproc0: powering up cm33
[  152.647167] remoteproc remoteproc0: Booting fw image rzg3s_cm33_rpmsg_linux-rtos_demo.elf, size
1068216
[  152.657050] remoteproc remoteproc0: unsupported resource 4
[  152.669150]  remoteproc0#vdev0buffer: assigned reserved memory node vdev0buffer@0x43200000
[  152.682507]  remoteproc0#vdev0buffer: registered virtio1 (type 7)
[  152.692313] remoteproc remoteproc0: remote processor cm33 is now up
```

Note: 1. **<device>** should be either rzg3s_cm33 or rzg3s_cm33-fpu.
2. **remoteprocX** can be either remoteproc0 or remoteproc1. When you would like to kick CM33, remoteproc0 should be specified. Also, remoteproc1 should be specified for kicking CM33_FPU.

4.4.5 CM33 Sample Program Invocation with BL2 of Trusted Firmware-A

On RZ/G3S SMARC EVK, you can invoke RPMsg sample program from BL2 of Trusted Firmware-A by the procedure stated below. Be sure to configure CA55 as boot CPU when enabling this feature.

1. Apply the following modification in red character to meta-renesas/meta-rzg3s/recipes-bsp/trusted-firmware-a/trusted-firmware-a.bbappend.

```
require trusted-firmware-a.inc

COMPATIBLE_MACHINE_rzg3s = "(rzg3s-dev|smarc-rzg3s)"

PLATFORM_rzg3s-dev = "g3s"
EXTRA_FLAGS_rzg3s-dev = "BOARD=dev14_1_lpddr PLAT_SYSTEM_SUSPEND=vbat"
EXTRA_FLAGS_smarc-rzg3s = "BOARD=smarc PLAT_SYSTEM_SUSPEND=vbat
PLAT_M33_BOOT_SUPPORT=1"
```

2. Rebuild Trusted Firmware-A.
3. Re-program the resultant BL2 and FIP binary files using FlashWriter.
4. Program rzg3s_cm33_rpmsg_demo.srec using FlashWriter with the parameter stated below:

- For Boot Mode 1

Partition Area	Start Address in sector	Program Start Address
1	1000	23000

- For Boot Mode 2

Address to load to RAM	Address to save to ROM
23000	200000

4.5 CA55 Sample Program Invocation

4.5.1 CA55 Sample Program Invocation for RZ/G2L, RZ/G2LC and RZ/G2UL

On RZ/G2L, G2LC and G2UL SMARC EVK, 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-<device> login: root
```

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

```
root@smarc-<device>:~# rpmsg_sample_client
```

4. Then, you can see the following message on the console associated with Pmod USBUART. Be sure that you invoke the 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/G2 CM33 ch0
2. communicate with RZ/G2 CM33 ch1

e. exit

please input
>
```

5. Then the number which performs the communication you would like to try on the console

4.5.2 CA55 Sample Program Invocation for RZ/G3S

On RZ/G3S SMARC EVK, 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-<device> login: root
```

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

```
root@smarc-<device>:~# rpmsg_sample_client
```

4. Then, you can see the following message on the console of Linux side.

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

1. communicate between CM33 cores
2. communicate between CM33 and CA55

e. exit

please input
>
```

5. Input the number which performs the communication you would like to try on the console. Please note that 1 is allowable ONLY when remoteproc support is enabled. Also, you must NOT invoke the CM33 program in advance. Meanwhile, in case of selecting 1, you need to invoke the CM33 program in advance.

6. In case of typing 1, the communication between CM33 cores should be established and the communication log is repeatedly displayed via the Pmod USBUART. Also, when 2 is typed, you can see the following message on Linux console:

```
[XXX] proc_id:0 rsc_id:0 mbx_id:0
metal: info:      metal_uio_dev_open: No IRQ for device 10400000.mbox-uio.
metal: info:      metal_uio_dev_open: No IRQ for device 11010000.cpg-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:0
[XXX] Initialize remoteproc successfully.
[XXX] proc_id:0 rsc_id:0 mbx_id:1
[XXX] Initialize remoteproc successfully.
[XXX] proc_id:1 rsc_id:1 mbx_id:1
[XXX] Initialize remoteproc successfully.

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

1. communicate with CM33 ch0
2. communicate with CM33 ch1
3. communicate with CM33_FPU ch0
4. communicate with CM33_FPU ch1
5. communicate with CM33 ch0 and CM33_FPU ch1

e. exit

please input
>
```

7. Input the number which performs the communication between CM33 and CA55 you would like to try on the console.

4.6 Overview of Sample Program Behavior

The behavior of RPMsg 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.

- **Termination message on CA55**

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

- **Termination message on CM33**

```
De-initializating remoteproc
```

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. Reference Documents

- R01AN5924: Getting Started with RZ/G Flexible Software Package
- R01US0553: RZ/G Verified Linux Package Version 3.0.6 Release Note
- R01US0616: SMARC EVK of RZ/G2L, RZ/G2LC, RZ/G2UL Linux Start-up Guide
- R01US0645: SMARC EVK of RZ/G3S Linux Start-up Guide

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Jul.30.21	-	First edition issued.
1.01	Nov.30.21	-	Updated in align with RZ/G2L BSP V1.3.
1.02	Jan.21.21	-	Updated in align with RZ/G2L BSP V1.3 update2.
1.10	Apr.27.22	-	Updated in align with RZ/G2L BSP V1.4.
1.11	May.30.22	-	Updated in align with RZ/G2 Verified Linux Package Version 3.0.0
1.12	Aug.31.22	-	Updated in align with RZ/G2 Verified Linux Package Version 3.0.0-update2
1.20	Nov.30.22	-	Updated in align with RZ/G2 Verified Linux Package Version 3.0.1
1.21	Apr.24.23	-	Updated in align with RZ/G2 Verified Linux Package Version 3.0.3
1.22	Oct 31.23	-	Updated in align with RZ/G2 Verified Linux Package Version 3.0.5
2.00	Jan.09.24	1, 3, 5-8, 11, 12, 14	Added RZ/G3S related description.
		13	CA55 console log stated in 4.6 was updated in accordance with the updates in latest RPMsg sample application.
2.01	Feb.13.24	-	Updated in align with RZ/G FSP Version 2.0.1.
		12	Fixed the address to write firmware to Flash ROM/eMMC.
2.0.2	Apr.24.24	-	Updated in align with RZ/G Verified Linux Package Version 3.0.6.
2.1.0	Aug.30.24	5-7	Added 3.5.2 in accordance with CM33 cold boot support.
		9-14	Added the procedure specific to CM33 cold boot support.
		18-19	Added 4.4.3 in accordance with Remoteproc support.
2.2.0	Oct.31.24	18	Added 4.4.4 in accordance with Remote support for RZ/G3S.
		21-22	Added 4.5.2 in accordance with the procedure specific to CA55 invocation for RZ/G3S.
2.3.0	Apr.25.25	-	Added AWO example project and update the package version to 2.3.0. For more information about the AWO example project, please refer to the included documentation.

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/.

© 2025 Renesas Electronics Corporation. All rights reserved.