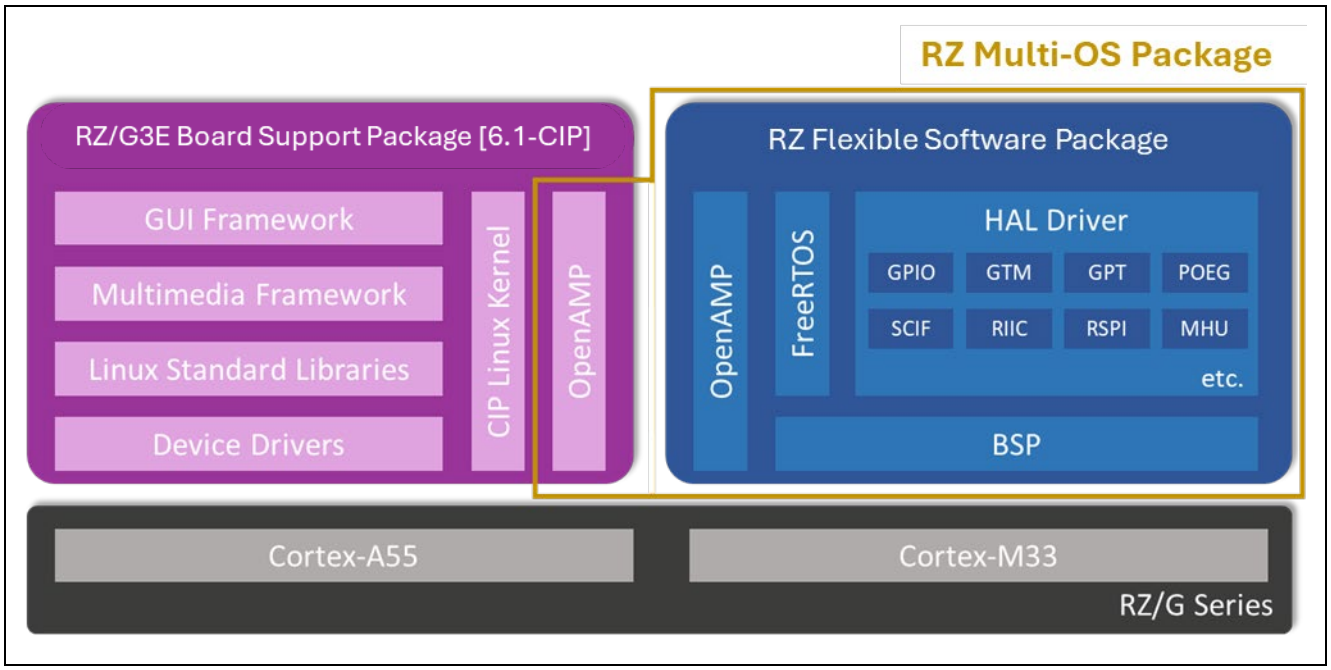
RZ/G3E

Quick Start Guide for RZ Multi-OS Package

Introduction

This document outlines the procedure for integrating the RZ Multi-OS Package into the RZ/G3E Board Support Package (referred to as BSP) for the RZ/G3E. By integrating the Multi-OS Package, users can efficiently establish a Multi-OS environment wherein Linux operates on the Cortex®-A55 and FreeRTOS/BareMetal runs on the Cortex-M33 with support for Inter-Processor Communication between these CPU cores.

This package requires the RZ Flexible Software Package (FSP) for an RTOS/BareMetal environment. The figure below illustrates the software stack of the RZ Multi-OS Package with the RZ/G3E:



Here are brief descriptions of each component related to RZ Multi-OS Package:

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

Target Device

RZ/G3E

Contents

1. Specifications

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

Table 1-1 Peripheral modules to be used in this package

Peripheral module	Usage
Message Handling Unit (MHU)	Configure Inter-Processor Interrupt.
Serial Communications Interface with FIFO (SCIFA)	Perform standard serial communications sending and receiving console messages.
Interrupt controller (INTC)	Handle the following types of interrupts as shown below for example: • Processors should receive interrupts during buffered serial communications. • MHU module fires Inter-Processor Interrupt.
General Purpose Input Output (GPIO)	Configure I/O lines used by serial communications.
General Timer (GTM)	Configure the tick for FreeRTOS.

2. Verified Operation Conditions

Table 2-1 shows the verified operation conditions.

Item	Contents
Integrated Development Environment	e^2 studio 2026-07 or later
Toolchain	GNU Arm Toolchain 13.3.Rel1 AArch32 bare-metal target (arm-none-eabi)
Dependent Software	• RZ Flexible Software Package (FSP) v4.2.0 • RZ/G3E Linux BSP v1.0.0

3. Sample Program Setup

3.1 Flexible Software Package Setup

Multi-OS Package expects RZ Flexible Software Package (FSP) to be installed in advance. For details on the installation, please refer to Getting Started with Flexible Software Package.

3.2 Integration of OpenAMP related stuff

This section describes how to integrate OpenAMP related stuff into RZ/G3E Linux BSP (hereinafter referred to as Linux BSP). The steps are based on **RZ/G3E-EVKIT Linux Start-up Guide** (hereinafter referred to as **Linux Start-up Guide**) included in **RZ/G3E Linux BSP v1.0.0**.

1. Follow the procedure stated from the beginning of **2.2 Building Images** to **(3) Add layers** of **Linux Start-up Guide**.
2. Download Multi-OS Package (r01an8260ej0420-rz-multi-os-pkg.zip) to a working directory and run the commands stated below:

```
cd ~/rzg3e_bsp_<pkg ver>
$ unzip <Multi-OS Dir>/r01an8260ej0420-rz-multi-os-pkg.zip
$ tar zxvf r01an8260ej0420-rz-multi-os-pkg/meta-rz-features_multi-os_v4.2.0.tar.gz
```

Here, <Multi-OS Dir> indicates the path to the directory where Multi-OS Package is placed.

3. Add layer for Multi-OS Package, simply run the helper script and select the **meta-rzg3e** layer from the list.

```
$ cd build
$ bash ../meta-rz-features/meta-rz-multi-os/add_meta_layer.sh
```

(Optional for remoteproc support)

4. Uncomment the following line in **meta-rz-features/meta-rz-multi-os/meta-rzg/meta-rzg3e/conf/layer.conf** for enabling remoteproc support:

```
MACHINE_FEATURES:append = " RZ_REMOTEPROC"
#MACHINE_FEATURES:append = " RZ_CM33_COLDBOOT"
#MACHINE_FEATURES:append = " RZ_CM33_FIRMWARE_LOAD"
#MACHINE_FEATURES:append = " RZ_CA55_CPU_CLOCKUP"
```

Note 1 – Requirement: It is required to uncomment the red lines, and it is required to keep the black lines commented.
Note 2 – Limitation: In this case, where CA55 cold boot is used and the Cortex-M33 is started via remoteproc, the CA55 clock-up option cannot be used.

5. Start a build as described in **(5) Start a build** of **2.2 Building Images** as shown below:

```
$ MACHINE=smarc-rzg3e bitbake core-image-<target>
```

For details on the allowable value of <target>, please refer to **Linux Start-up Guide**.

3.3 Note for Integration

By default, all clock-related configuration is handled on the Linux (main core) side. If the system design is changed such that clocks for specific drivers are configured by the RTOS (sub core) side, the implementation must be customized to ensure these settings do not conflict with Linux's clock initialization sequence.

The peripherals which are NOT enabled enter Module Standby Mode after Linux kernel is booted up. That means the peripherals used on sub core (CM33) side might stop working at that time. To avoid such a situation, Multi-OS Package incorporates the patch below:

- 0002-Set-GTM-as-critical-clock-to-support-RTOS.patch

This patch prevents GTM used in RPMsg demo program from entering Module Standby Mode. If you have any other peripherals which you would like to stop entering Module Standby implicitly, please update the patch:

```
diff --git a/drivers/clk/renesas/r9a09g047-cpg.c b/drivers/clk/renesas/r9a09g047-cpg.c
index c5c4c55ce7a5..b9668c2c058a 100644
--- a/drivers/clk/renesas/r9a09g047-cpg.c
+++ b/drivers/clk/renesas/r9a09g047-cpg.c
@@ -328,14 +328,14 @@ static const struct rzv2h_mod_clk r9a09g047_mod_clks[] = {
  DEF_MOD("rcpu_cmtw1_clkm",    CLK_PLLCLN_DIV32, 4, 0, 2, 0),
  DEF_MOD("rcpu_cmtw2_clkm",    CLK_PLLCLN_DIV32, 4, 1, 2, 1),
  DEF_MOD("rcpu_cmtw3_clkm",    CLK_PLLCLN_DIV32, 4, 2, 2, 2),
- DEF_MOD("gtm_0_pclk",         CLK_PLLCM33_DIV16, 4, 3, 2, 3),
- DEF_MOD("gtm_1_pclk",         CLK_PLLCM33_DIV16, 4, 4, 2, 4),
- DEF_MOD("gtm_2_pclk",         CLK_PLLCLN_DIV16, 4, 5, 2, 5),
+ DEF_MOD_CRITICAL("gtm_0_pclk",       CLK_PLLCM33_DIV16, 4, 3, 2, 3),
+ DEF_MOD_CRITICAL("gtm_1_pclk",       CLK_PLLCM33_DIV16, 4, 4, 2, 4),
+ DEF_MOD_CRITICAL("gtm_2_pclk",       CLK_PLLCLN_DIV16, 4, 5, 2, 5),
  DEF_MOD("gtm_3_pclk",         CLK_PLLCLN_DIV16, 4, 6, 2, 6),
  DEF_MOD("gtm_4_pclk",         CLK_PLLCLN_DIV16, 4, 7, 2, 7),
- DEF_MOD("gtm_5_pclk",         CLK_PLLCLN_DIV16, 4, 8, 2, 8),
- DEF_MOD("gtm_6_pclk",         CLK_PLLCLN_DIV16, 4, 9, 2, 9),
- DEF_MOD("gtm_7_pclk",         CLK_PLLCLN_DIV16, 4, 10, 2, 10),
+ DEF_MOD_CRITICAL("gtm_5_pclk",       CLK_PLLCLN_DIV16, 4, 8, 2, 8),
+ DEF_MOD_CRITICAL("gtm_6_pclk",       CLK_PLLCLN_DIV16, 4, 9, 2, 9),
```

```
+ DEF_MOD_CRITICAL("gtm_7_pclk",       CLK_PLLCLN_DIV16, 4, 10, 2, 10),
  DEF_MOD("wdt_0_clkp",          CLK_PLLCM33_DIV16, 4, 11, 2, 11),
  DEF_MOD("wdt_0_clk_loco",      CLK_QEXTAL, 4, 12, 2, 12),
  DEF_MOD("wdt_1_clkp",          CLK_PLLCLN_DIV16, 4, 13, 2, 13),
```

Replacing the DEF_MOD of the target clock with a DEF_MOD_CRITICAL patch prevents the Linux BSP from changing the module to standby mode.

3.4 Deployment of RZ/G3E Linux BSP

With respect to the deployment of Linux kernel, device tree and root filesystem for RZ/G3E, please refer to **Linux Start-up Guide**.

4. Sample Program Invocation on RZ/G3E

4.1 Hardware setup

1. Connect J-Link to RZ/G3E SMARC EVK. For details, please refer to Getting Started with Flexible Software Package.

4.2 CM33 Sample Program Setup

Here are the procedures for setting up the sample program running on CM33.

1. Extract **r01an8260ej0420-rz-multi-os-pkg.zip** on your development PC.
2. Extract **rzg3e_cm33_rpmsg_linux_rtos_example.zip** included in **r01an8260ej0420-rz-multi-os-pkg.zip**.
3. Open e^2 studio 2026-07 and click **File** > **Import**.
4. Double-click General and select **Existing Projects into Workspace** as shown in Figure 4-1:

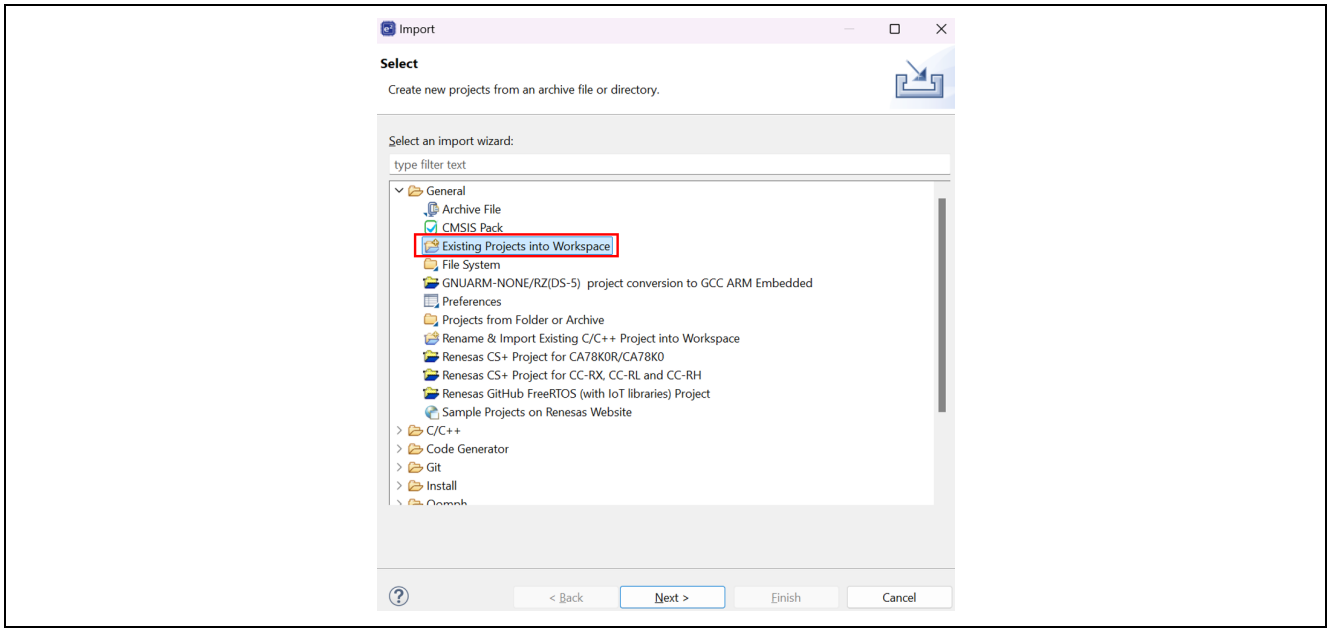


Figure 4-1. Import sample project (1)

5. Input the path to the directory where sample project you would like to import to **Select root directory**, press **Enter** key and click **Finish** button.

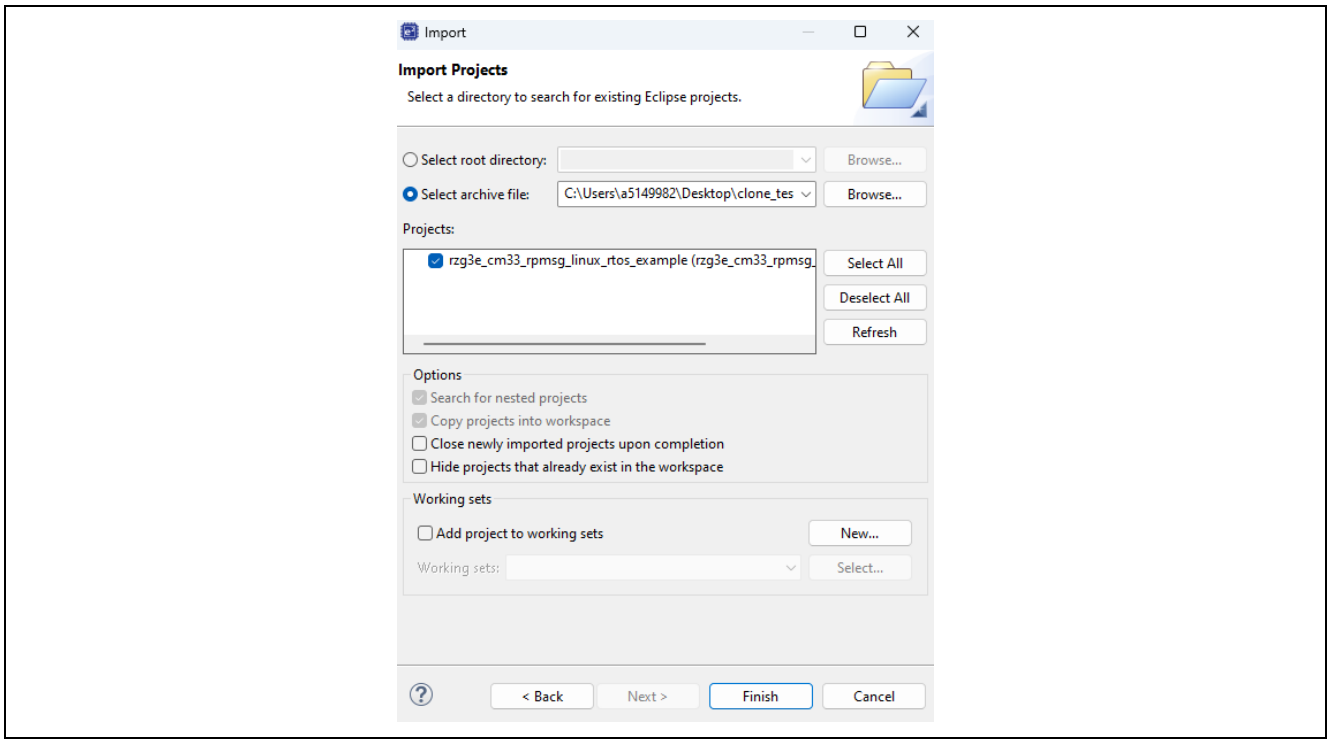


Figure 4-2. Import sample project (2)

(Optional)

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

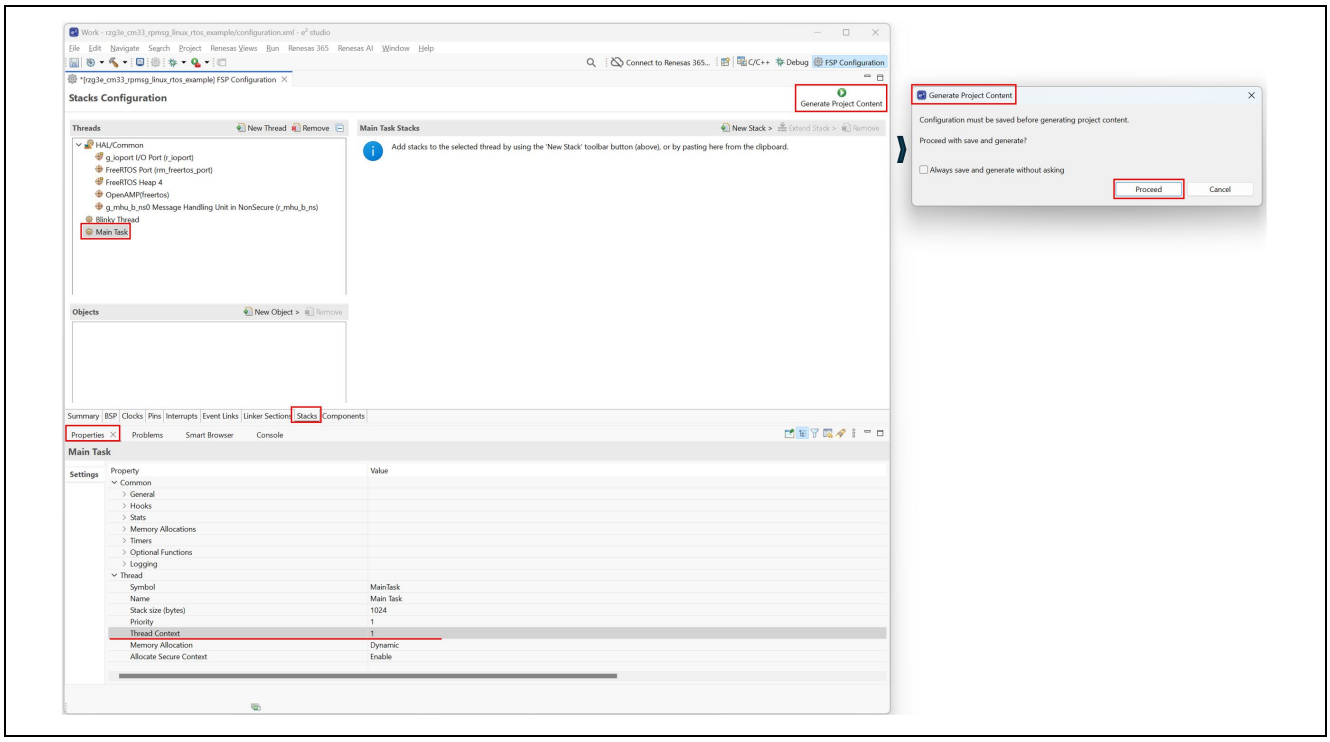


Figure 4-3. RPMsg Channel Setting

(Optional for remoteproc support)

7. Change the value for **ENABLE_REMOTEPROC** defined in **platform_info.h** from 0 to 1 as shown below:

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

8. Build the project from **Choose Project** > **Build Project**.
9. If building project is successfully completed, build artifacts as listed below should be generated in **Debug** or **Release** directory of the project you imported in accordance with the active Build Configuration.
   - rzg3e_cm33_rpmsg_linux_rtos_example.elf
   - rzg3e_cm33_rpmsg_linux_rtos_example.bin

4.3 CM33 Sample Project Invocation for communicating with Linux

4.3.1 CM33 Sample Project Invocation using Segger J-Link

Carry out the procedure shown below for invoking CM33 sample project using J-Link:

1. Click Debug button as shown below:

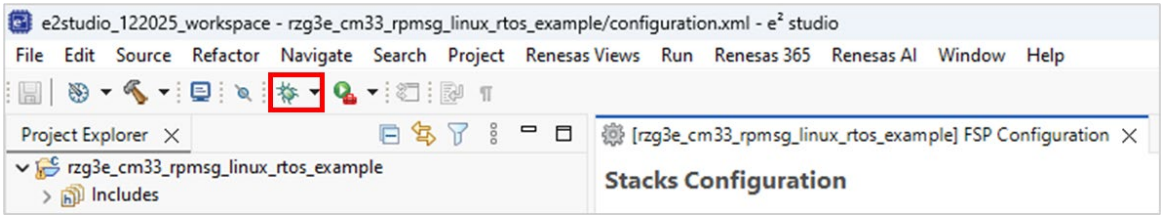


Figure 4-4. Selection of Debug Configuration

2. If the following **Select Configuration** window is shown, choose rzg3e_cm33_rpmsg_linux_rtos_example_Debug_Flat or

rzg3e_cm33_rpmsg_linux_rtos_example_Release_Flat in accordance with the build configuration you are using and click **OK**.

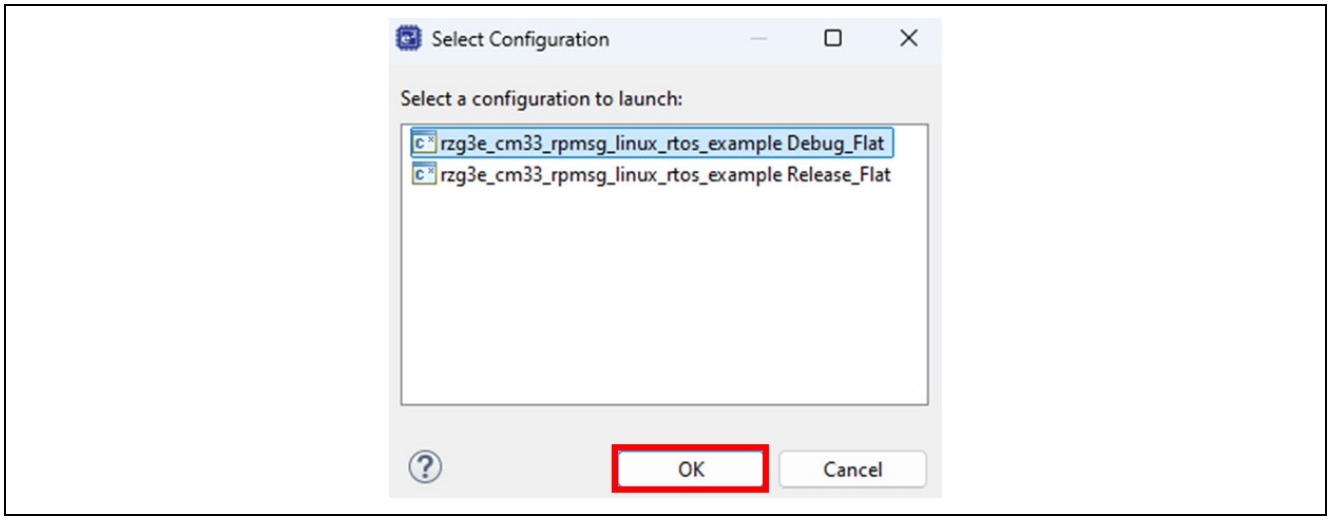


Figure 4-5. Debug Perspective Launch (1)

If the following **Confirm Perspective Switch** window appears, press **Switch** to go ahead.

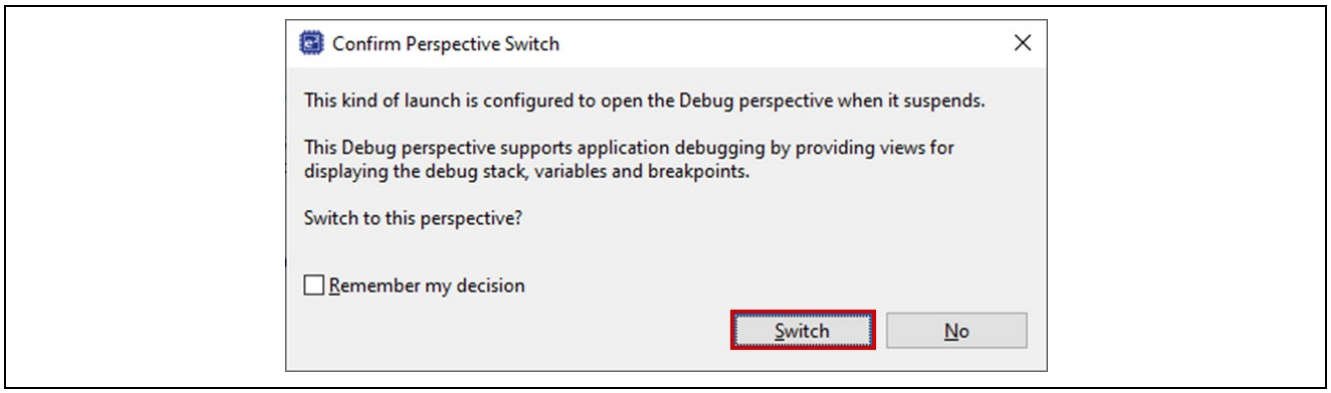


Figure 4-6. Debug Perspective Launch (2)

3. When **Debug Perspective** is opened, Program Counter (PC) should be located at the top of **Entry_Function_S**. Then, press the button shown in Figure 4.7.

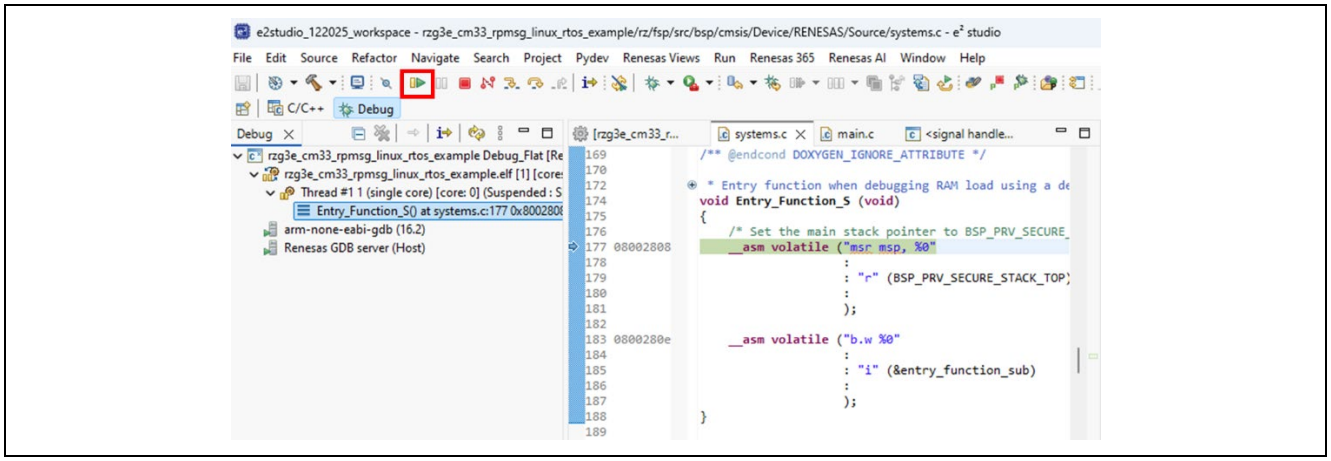


Figure 4-7. How to start to debug Sample Project (1)

4. PC should be stopped at the top of **main** function. Then, click the same button in the previous step to continue.

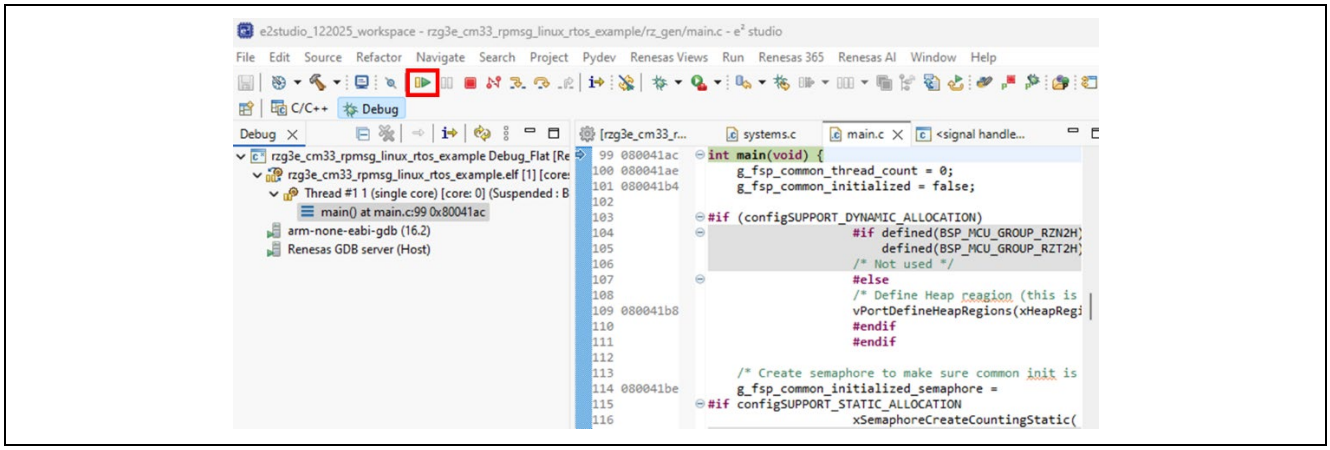


Figure 4-8. How to start to debug Sample Project (2)

5. CM33 Sample Project starts to work and wait for the launch of CA55 RPMsg Sample Program.

4.3.2 CM33 Sample Program Invocation from u-boot

Here is the procedure for invoking CM33 Sample Program from u-boot:

1. Place rzg3e_cm33_rpmsg_linux_rtos_example.bin together with Linux kernel and Device Tree Blob.
2. Insert SD card into SD0 of RZ/G3E SMARC EVK.
3. Turn on RZ/G3E SMARC EVK by pressing the Power button for a few seconds.
4. Hit any key to stop autoboot within 3 seconds after the following message on the console associated with SER3_UART of RZ/G3E SMARC EVK.

```
U-Boot 2024.07 (Jun 13 2025 - 15:56:19 +0000)

CPU:   Renesas Electronics CPU rev 1.0
Model: Renesas SMARC EVK based on r9a09g047e57
DRAM:  3.9 GiB
Core:  45 devices, 18 uclasses, devicetree: separate
MMC:   mmc@15c00000: 0, mmc@15c10000: 1, mmc@15c20000: 2
Loading Environment from MMC... Reading from MMC(0)... OK
In:    serial@11c01400
Out:   serial@11c01400
Err:   serial@11c01400
Net:
Error: ethernet@15C40000 No valid MAC address found.

Error: ethernet@15C30000 No valid MAC address found.

Error: ethernet@15C30000 No valid MAC address found.

Error: ethernet@15C40000 No valid MAC address found.
No ethernet found.

Hit any key to stop autoboot:  0
=>
```

4. Carry out the following setup on u-boot to kick 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/rzg3e_cm33_rpmsg_linux_rtos_example.bin;
mw.l 0x10420C0C 0x00000000; dcache on'
=> saveenv
=> run cm33start

Note: The above is the command for eSD boot mode. If you want to use this
command in SPI flash or eMMC boot mode, replace the red text above with "1:2".
```

4.3.3 CM33 Sample Program Invocation with remoteproc

Here is the procedure for invoking CM33 Sample Program with remoteproc:

1. Booting up Linux by following **Linux Start-up Guide**.
2. Invoke the command stated below to specify the sample program to be loaded:

```
root@smarc-rzg3e:~# echo rzg3e_cm33_rpmsg_linux_rtos_example.elf >
/sys/class/remoteproc/remoteproc0/firmware
```

3. Kick CM33 by invoking the command below:

```
root@smarc-rzg3e:~# echo start > /sys/class/remoteproc/remoteproc0/state
```

If CM33 Sample Program starts to work successfully, the following message should be shown on Linux console:

```
root@smarc-rzg3e:~# echo start > /sys/class/remoteproc/remoteproc0/state
[   45.290143] remoteproc remoteproc0: powering up cm33
[   45.377860] remoteproc remoteproc0: Booting fw image
rzg3e_cm33_rpmsg_linux_rtos_example.elf, size 1330072
[   45.390858] rproc-virtio rproc-virtio.1.auto: assigned reserved memory node
vdev0buffer@0x43200000
[   45.399950] rproc-virtio rproc-virtio.1.auto: registered virtio0 (type 7)
[   45.406751] remoteproc remoteproc0: remote processor cm33 is now up
```

Note: The firmware components are stored in the file system at /lib/firmware folder. Please replace it if any updates to firmware.

4.3.4 Note for Suspend-to-RAM behavior

When the system resumes from a Suspend-to-RAM process, be aware of the following behavior:

1. The CM33 core is shut down.
2. **In the U-boot invocation method**: The CM33 firmware cannot be restarted via U-Boot because U-Boot is not restarted during the resume process.
3. **In the Remoteproc invocation method**:The firmware must be stopped first using the remoteproc command before the CM33 firmware can be restarted. You can do this with the following commands:

```
root@smarc-rzg3e:~# echo stop > /sys/class/remoteproc/remoteproc0/state
root@smarc-rzg3e:~# echo rzg3e_cm33_rpmsg_linux_rtos_example.elf >
/sys/class/remoteproc/remoteproc0/firmware
root@smarc-rzg3e:~# echo start > /sys/class/remoteproc/remoteproc0/state
```

4.4 CA55 Sample Program Invocation

This section describes how to invoke RPMsg sample program running on CA55 side.

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

```
=> run bootcmd
```

2. Login as **root**.

```
smarc-rzg3e login: root
```

3. Invoke RPMsg sample program as shown below:

```
root@smarc-rzg3e:~# rpmsg_sample_client
```

4. If the sample program started to work successfully, you can see the following message on Linux console:

```
metal: info: metal_uio_dev_open: No IRQ for device 10480000.mbox-uio.
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/G3E CM33 ch0
2. communicate with RZ/G3E CM33 ch1
e. exit
please input
>
```

5. Type **1** if RPMsg channel 0 is used in the CM33 RPMsg sample program under the default settings. Otherwise, type **2** to start communication using RPMsg channel 1.
6. By typing e, the 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
```

4.5 Overview of Sample Program's behavior

This section describes the overview of sample program's behavior.

1. When the CA55 sample program is successfully executed, the communication channel between CA55 and CM33 is established.
2. The CA55 sample program starts to send the message to CM33 with incrementing the message size from the minimum value 17 to the maximum value 488. During the communication, the message as shown below is displayed on your console:

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

3. When CM33 sample program receives the message sent from CA55, the echo reply is sent back to CA55 sample program.
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 488-byte sized payload is sent from CA55 to CM33 and CM33 sends back the echo reply, the message indicating the termination of the communication channel is sent from CA55 to CM33. Then, the CA55 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. CA55 1.8GHz configuration support at CA55 cold boot mode

This chapter describes how to configure 1.8GHz as operational frequency of CA55 at CA55 cold boot.

5.1 Setup of CA55 related stuff

1. Uncomment the following lines in meta-rz-features/meta-rz-multi-os/meta-rzg/meta-rzg3e/conf/layer.conf

```
#MACHINE_FEATURES:append = " RZ_REMOTEPROC"
#MACHINE_FEATURES:append = " RZ_CM33_COLDBOOT"
MACHINE_FEATURES:append = " RZ_CM33_FIRMWARE_LOAD"
MACHINE_FEATURES:append = " RZ_CA55_CPU_CLOCKUP"
```

Note 1 – Requirement: It is required to uncomment the red lines, and it is required to keep the black lines commented.
Note 2 – Limitation: In this case, where CA55 cold boot is used and the Cortex-M33 has already been started from BL2 for clock up Cortex-A55, remoteproc cannot be used.

2. Rebuild TrustedFirmware-A and Linux Kernel as shown below:

```
MACHINE=smarc-rzg3e bitbake trusted-firmware-a -c cleansstate
MACHINE=smarc-rzg3e bitbake firmware-pack -c cleansstate
MACHINE=smarc-rzg3e bitbake linux-renesas -c cleansstate
MACHINE=smarc-rzg3e bitbake core-image-weston
```

3. Deploy build artifacts to SD card by following **Linux Start-up Guide**.

5.2 Setup of CM33 related stuff

1. Create RZ FSP project for CM33.
   When creating a project, select the board that is dedicated to this function, as highlighted below.

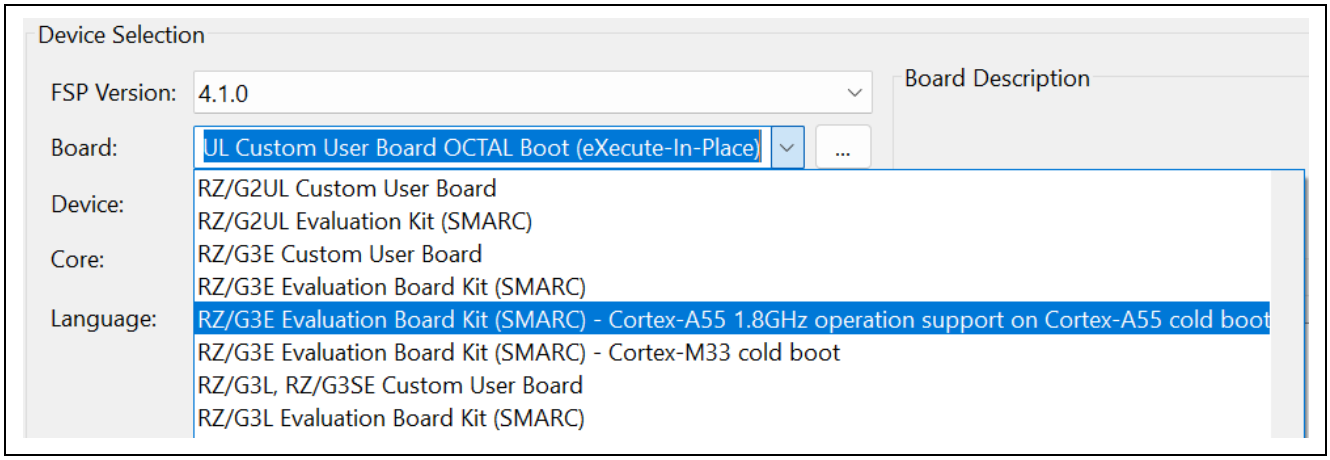


Figure 5-1. Select board for CA55 1.8GHz operation

2. Open **configuration.xml** in the project and choose **BSP** tab.
3. Configure **Clock up for CA55** properties as Enable. Also, enabled **Launch CA55(core0)** if you would like to configure operational at CM33 cold boot mode.

4. Click **Generate Project Content** to reflect the changes to your project.

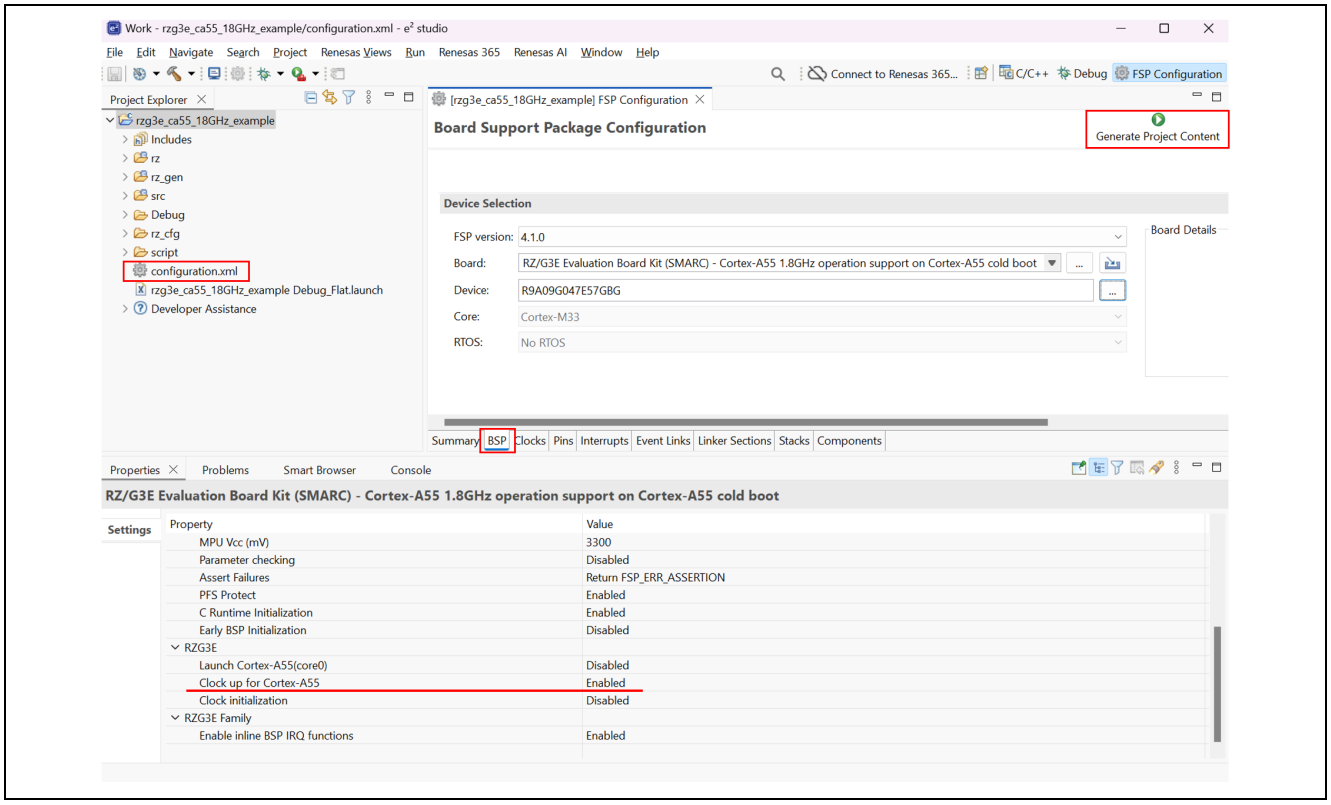


Figure 5-2. CM33 project setting for CA55 1.8GHz support (CA55 coldboot)

6. Build the project from **Project** > **Build Project**.

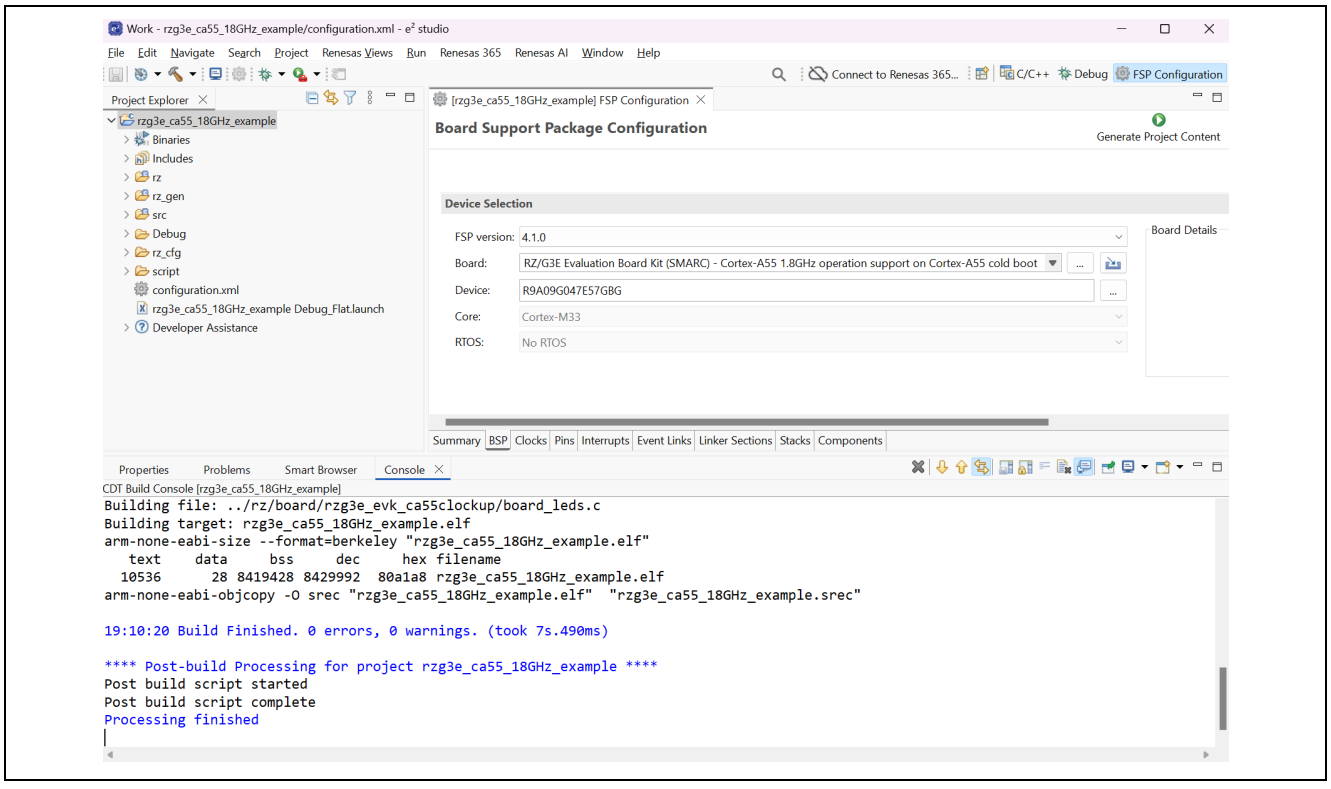


Figure 5-3. Build CM33 project for CA55 1.8GHz support

5.3 Deployment of Build Artifacts to SMARC RZ/G3E

This section describes the procedure to program the build artifacts to SMARC RZ/G3E.

1. Connect SER3_UART of SMARC RZ/G3E with Host PC and established serial port connection.
2. Configure SW_MODE-1, SW-MODE-2 and SW_MODE-3 of SMARC RZ/G3E as OFF, ON and OFF respectively to specify boot mode as SCIF download mode.
3. Turn on SMARC RZ/G3E. Then, the following message is shown on your terminal:

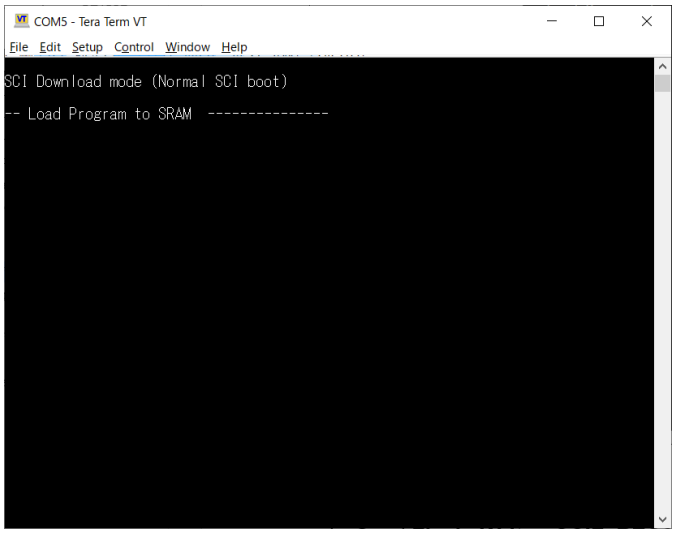


Figure 5-4. SCIF Download mode

4. Send **Flash_Writer_SCIF_RZG3E_EVK_LPDDR4X.mot** to SMARC RZ/G3E via terminal software. If it's successfully transferred, the following message is shown on your terminal:

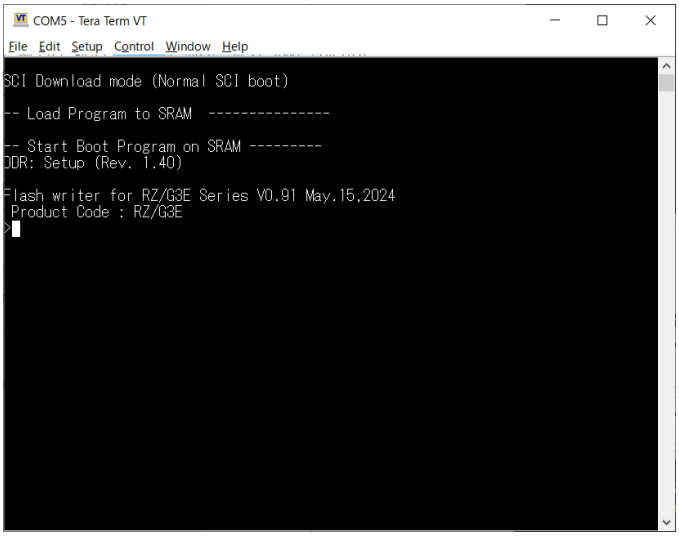


Figure 5-5. Flash Writer invocation

5. Program **bl2_bp_spi-smarc-rzg3e.srec** with Flash Writer as shown below:

```
xls2
===== Qspi writing of RZ/G2 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'8003600

===== Please Input Qspi Save Address ===
  Please Input : H'00000
please send ! ('.' & CR stop load)
Erase SPI Flash memory...
Erase Completed
Write to SPI Flash memory.
======= Qspi  Save Information  =================
SpiFlashMemory Stat Address : H'00000000
SpiFlashMemory End Address  : H'00037E57
============================================================
```

6. Program **fip-smarc-rzg3e.srec** with Flash Writer as shown below:

```
xls2
===== Qspi writing of RZ/G2 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'00000

===== Please Input Qspi Save Address ===
  Please Input : H'60000
please send ! ('.' & CR stop load)
Erase SPI Flash memory...
Erase Completed
Write to SPI Flash memory.
======= Qspi  Save Information  =================
SpiFlashMemory Stat Address : H'00060000
SpiFlashMemory End Address  : H'0011C2BE
============================================================
```

7. Program CM33 FW (S-record formatted one) with Flash Writer as shown below:

```
xls2
===== Qspi writing of RZ/G2 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'00000

===== Please Input Qspi Save Address ===
  Please Input : H'202000
please send ! ('.' & CR stop load)
Erase SPI Flash memory...
Erase Completed
Write to SPI Flash memory.
======= Qspi  Save Information  =================
SpiFlashMemory Stat Address : H'00202000
SpiFlashMemory End Address  : H'002071F2
============================================================
```

6. CM33 cold boot support

This chapter describes how CA55 and CM33 related stuff should be built for CM33 cold boot.

6.1 Setup of CA55 related stuff

1. Uncomment the following lines in meta-rz-features/meta-rz-multi-os/meta-rzg/meta-rzg3e/conf/layer.conf

```
#MACHINE_FEATURES:append = " RZ_REMOTEPROC"
MACHINE_FEATURES:append = " RZ_CM33_COLDBOOT"
#MACHINE_FEATURES:append = " RZ_CM33_FIRMWARE_LOAD"
#MACHINE_FEATURES:append = " RZ_CA55_CPU_CLOCKUP"
```

Note 1 – Requirement: It is required to uncomment the red lines, it is required to keep the black lines commented, and the blue lines are optional and may be modified.
Note 2 – Limitation: In this case, where CM33 cold boot is used and the Cortex-M33 had already been started, remoteproc cannot be used.

2. Rebuild TrustedFirmware-A as shown below:

```
MACHINE=smarc-rzg3e bitbake trusted-firmware-a -c cleansstate
MACHINE=smarc-rzg3e bitbake firmware-pack -c cleansstate
MACHINE=smarc-rzg3e bitbake core-image-weston
```

3. Deploy build artifacts to SD card by following **Linux Start-up Guide**.

6.2 Deployment of CA55 Build Artifacts to SMARC RZ/G3E

This section describes how to deploy CA55 Build Artifacts to SMARC RZ/G3E. First, please carry out the same procedure as 1 to 4 stated in **5.3 Deployment of Build Artifacts to SMARC RZ/G3E**. Then, follow the steps stated below:

1. Program **bl2_bp_spi-smarc-rzg3e.srec** with Flash Writer as shown below:

```
xls2
===== Qspi writing of RZ/G2 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'8003600

===== Please Input Qspi Save Address ===
  Please Input : H'100000
please send ! ('.' & CR stop load)
Erase SPI Flash memory...
Erase Completed
Write to SPI Flash memory.
======= Qspi  Save Information  =================
SpiFlashMemory Stat Address : H'00000000
SpiFlashMemory End Address  : H'00036D17
============================================================
```

2. Program **fip-smarc-rzg3e.srec** with Flash Writer as shown below:

```
xls2
===== Qspi writing of RZ/G2 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'00000

===== Please Input Qspi Save Address ===
  Please Input : H'160000
please send ! ('.' & CR stop load)
Erase SPI Flash memory...
Erase Completed
Write to SPI Flash memory.
======= Qspi  Save Information  =================
SpiFlashMemory Stat Address : H'00280000
SpiFlashMemory End Address  : H'0033C2BE
============================================================
```

6.3 Setup of CM33 related stuff

2. Create RZ FSP project for CM33.
   When creating a project, select the board that is dedicated to this function, as highlighted below.

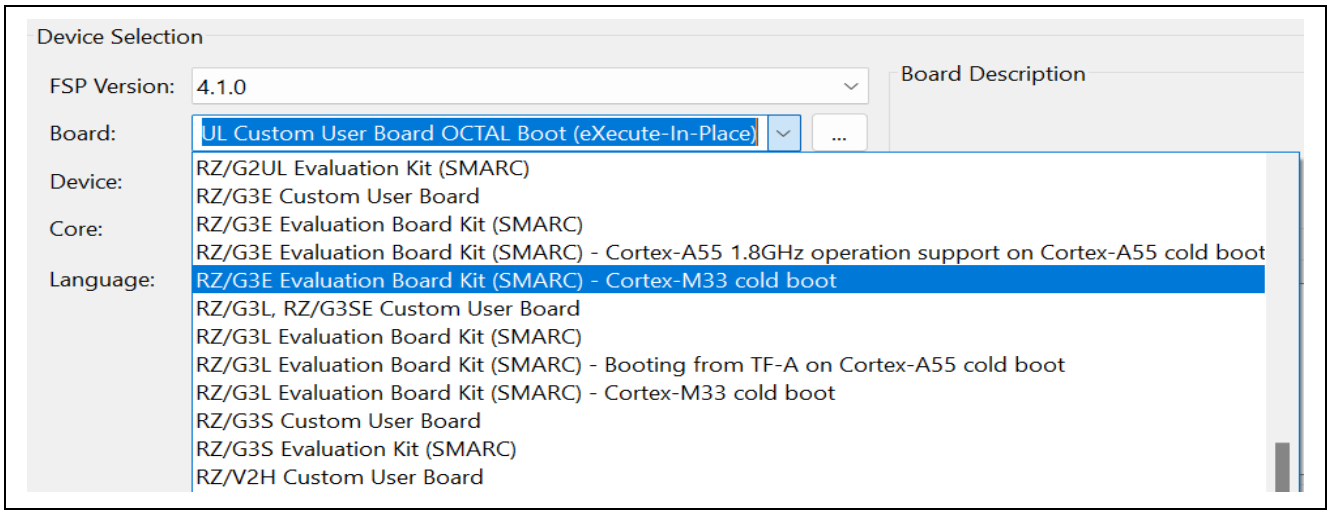


Figure 6-1. Select board for CM33 cold boot

2. Open **configuration.xml** in the project and choose **BSP** tab.
3. Make sure **Launch Cortex-A55 (core0)** is Enabled.
4. (Optional) Configure **Clock up for Cortex-A55** as Enabled.
5. Click **Generate Project Content** to reflect the changes to your project.

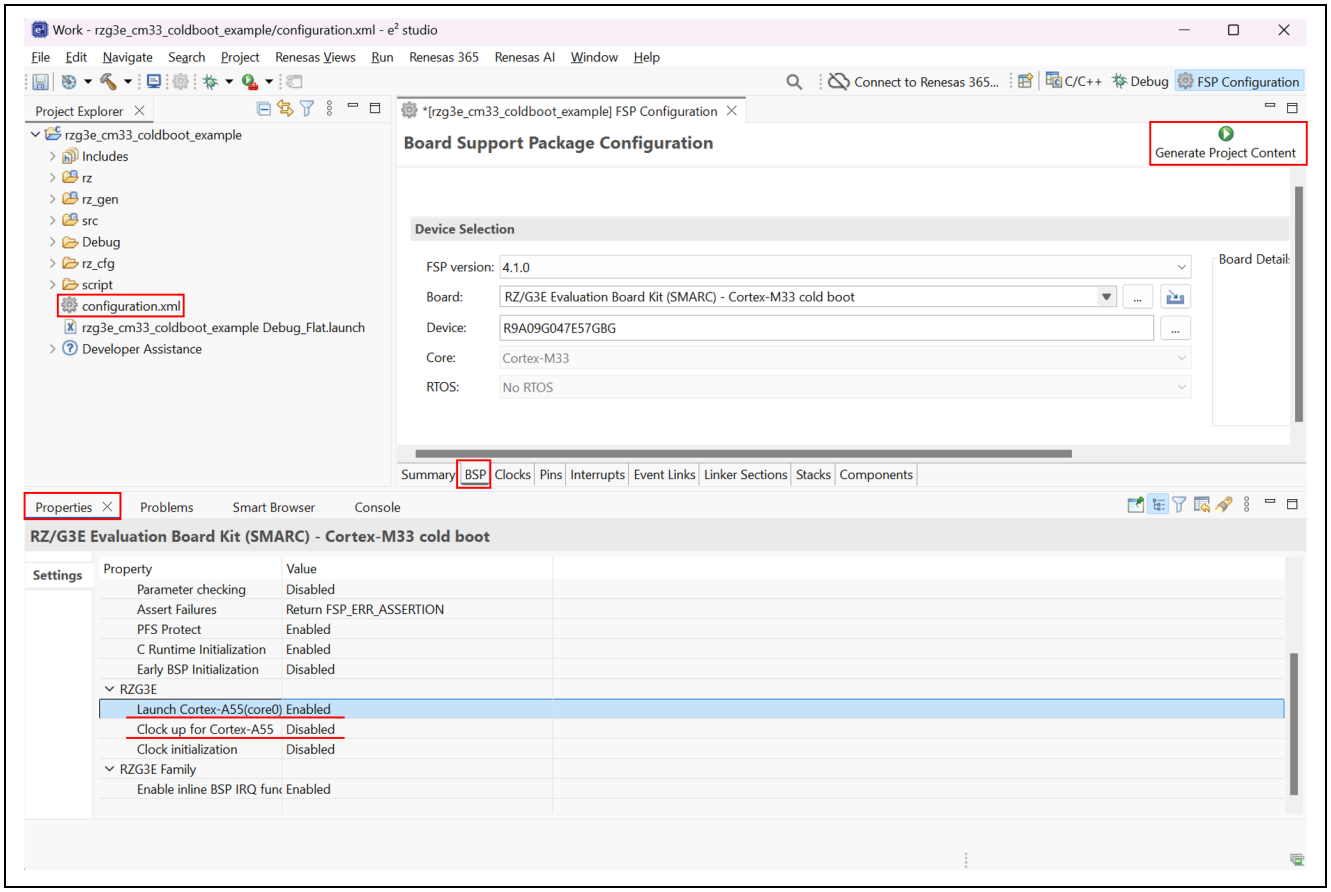


Figure 6-2. CM33 project setting for CM33 cold boot

6. Build the project from **Project** > **Build Project**.
7. Click **Run** > **Debug Configurations...** , expand **Renesas GDB Hardware Debugging** and choose **<project name> Debug_Flat**.

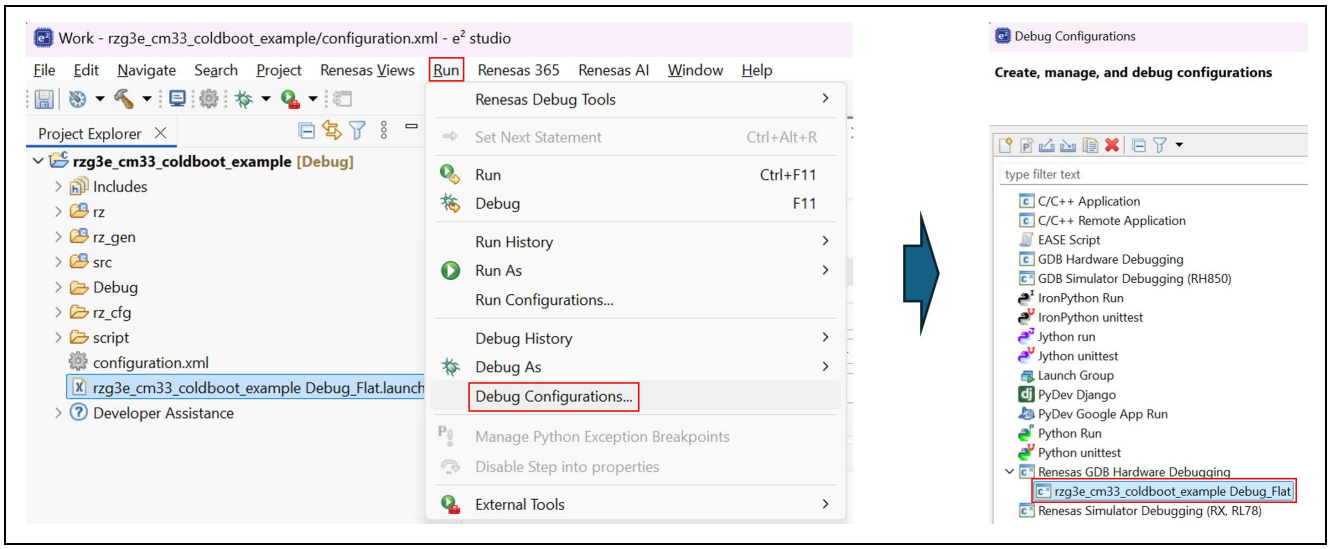


Figure 6-3. Debug Configuration Launch

8. Choose **Debugger** > **Connection Settings** and specify **Yes** to **Reset after download**.

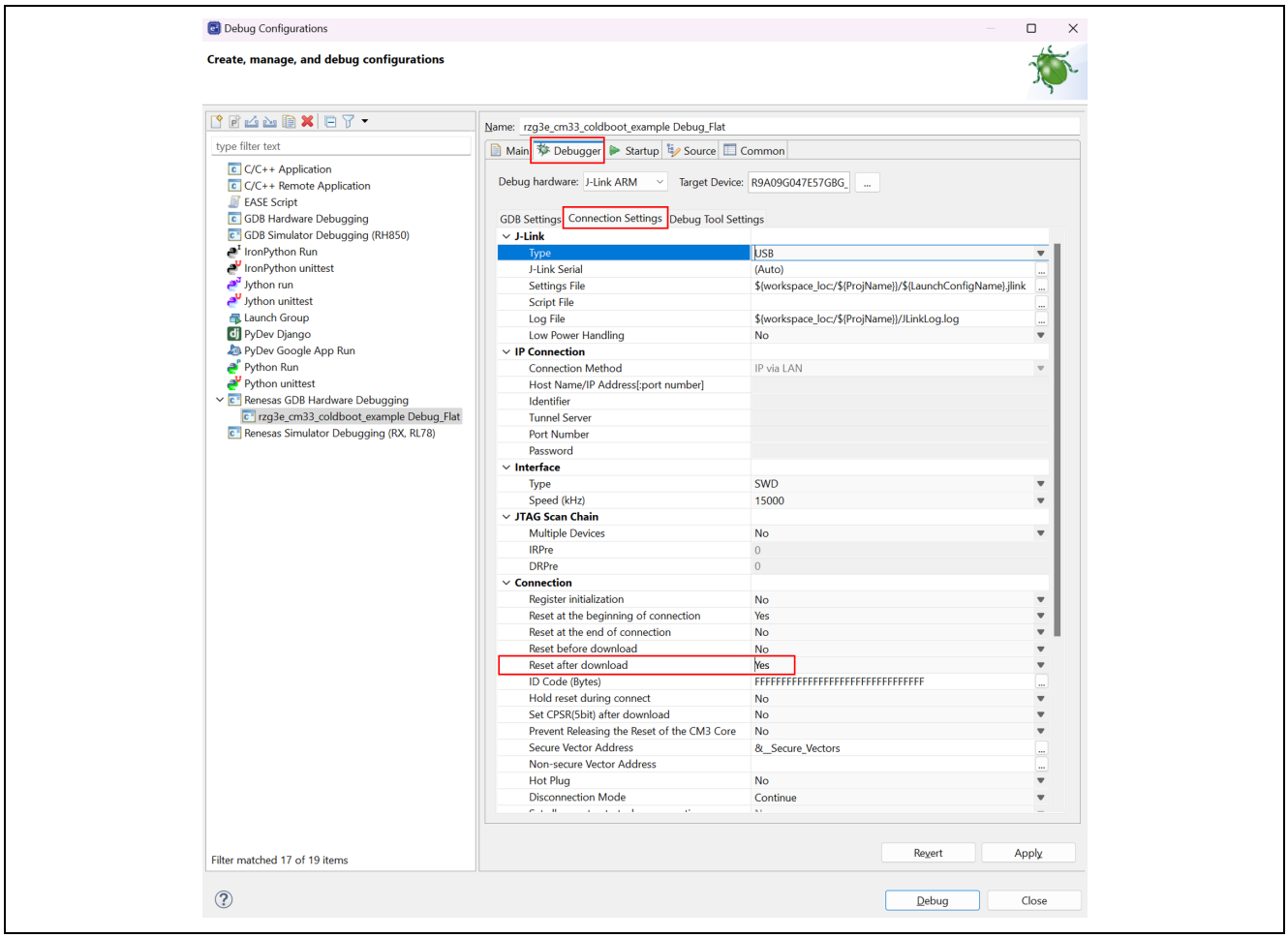


Figure 6-4. Connection Settings

9. Choose Startup and change the Load type of Program Binary to Symbols only.

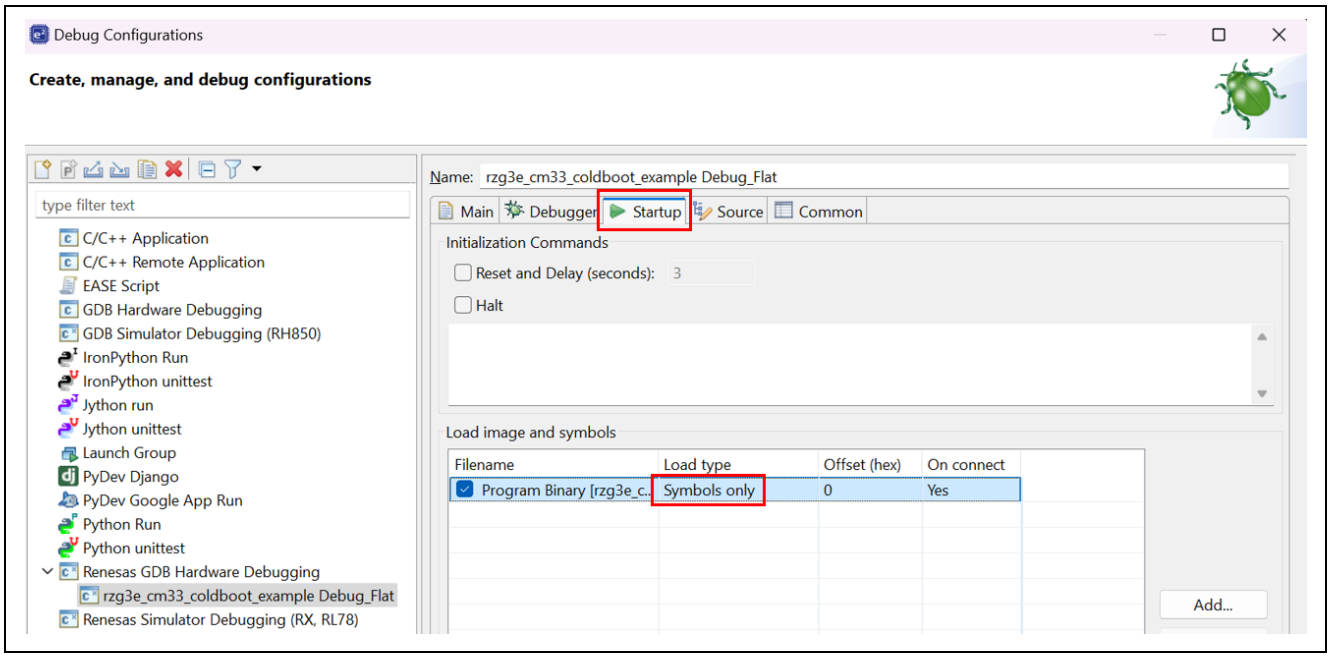


Figure 6-5. Startup Settings (1)

10. Click Add... > Workspace… , choose SREC file and click OK.

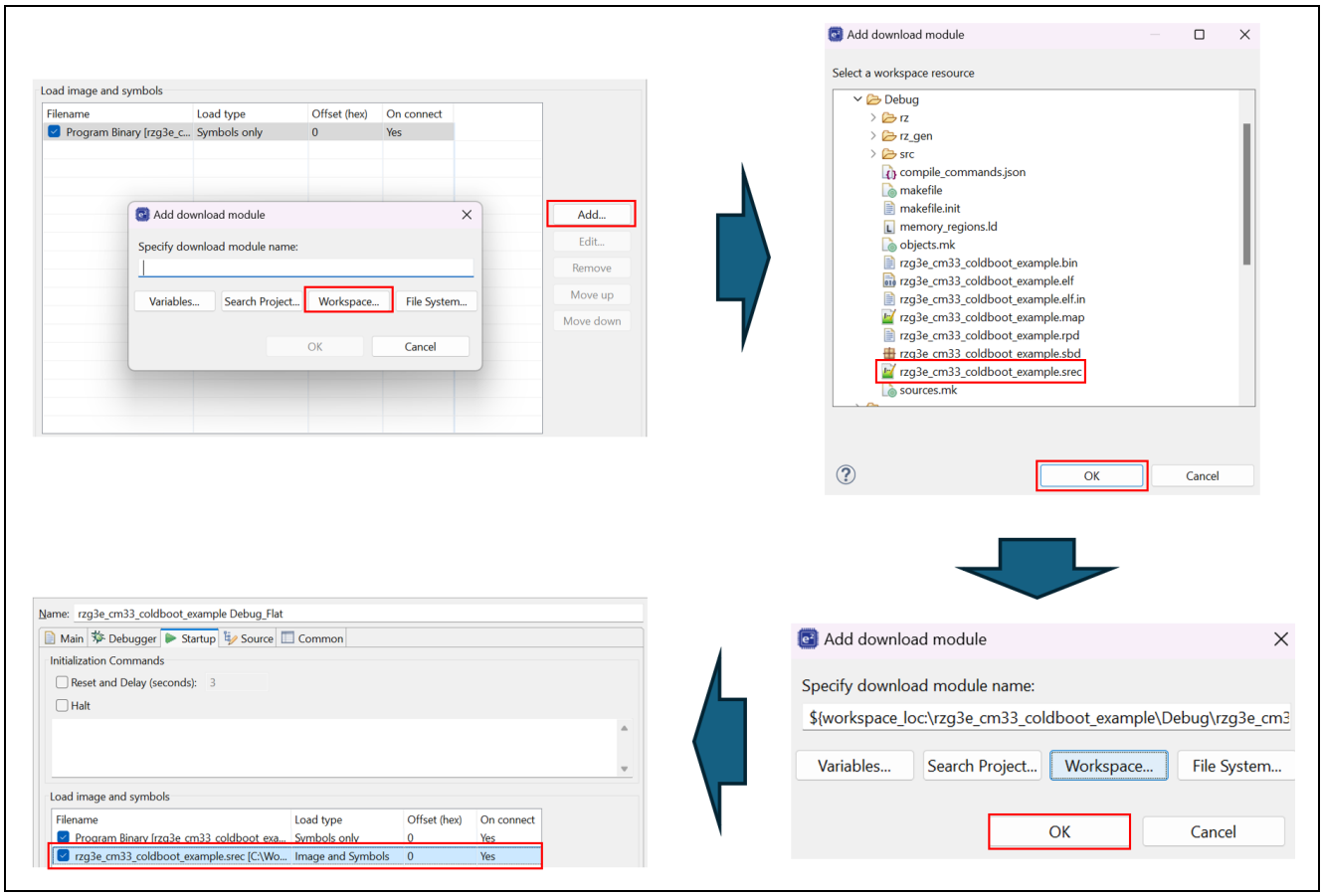


Figure 6-6. Startup Settings (2)

11. Click **Debug** to launch Debug Perspective. In addition, it is necessary to change the DIPSW setting on the board to the following CM33 cold boot and SPI boot mode settings in advance. Then, CM33 program starts, load CA55 build artifacts from QSPI Flash ROM and kick CA55.

BOOT

	1	2	3	4	5	6
ON		■				
OFF	■		■	■	■	■

SW_MODE

	1	2	3	4
ON				■
OFF	■	■	■	

For details on CM33 program invocation with e2studio, please see Getting Started with Flexible Software Package.

7. Assignment of peripherals

Table 7.1 shows the expected assignment of peripherals to sub core (CM33) on this example project while all other devices remain assigned to the Linux (main core) side.

Table 7.1 Peripherals assignment on RZ/G3E Multi-OS Example Project

Peripherals	**CPU**		**Remarks**
	Main core CA55	**Sub core CM33**	
GTM	ch2-3	ch0, ch1, ch5, ch6, ch7	0002-Set-GTM-as-critical-clock-to-support-RTOS.patch

8. Appendix

8.1 Setting for RPMsg Example supporting OpenAMP v2024.05

For support OpenAMP v2024.05, the RPMsg examples require additional preprocessor to build and run successfully with the updated virtio/OpenAMP configuration as below:

`VIRTIO_DRIVER_SUPPORT=1` for the Master core.
`VIRTIO_DEVICE_SUPPORT=1` for the Slave core.

Configuration path:
e² studio (GCC): *Properties → Cross ARM C Compiler → Preprocessor → Defined symbols*

9. Reference Documents

- R01AN5924 RZ/G2L RZ/G2LC RZ/G2UL RZ/G3S RZ/G3E Getting Started with Flexible Software Package
- R01UH1145 RZ/G3E SMARC Module Board User's Manual: Hardware
- R01UH1077 RZ SMARC Series Carrier Board II User's Manual: Hardware

Revision History

Rev.	Date	Description	
		Page	Summary
3.00	Jul.22.2025	-	First edition.
3.10	Dec.26.2025	-	Update Multi-OS Package version to 3.1.0.
3.20	Feb.06.2026	-	Update Multi-OS Package version to 3.2.0.
4.00	Mar.31.2026	-	Update Multi-OS Package version to 4.0.0.
4.10	Jun.30.2026	-	Update Multi-OS Package version to 4.1.0.
4.20	Sep.03.2026	-	Update Multi-OS Package version to 4.2.0.

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/.