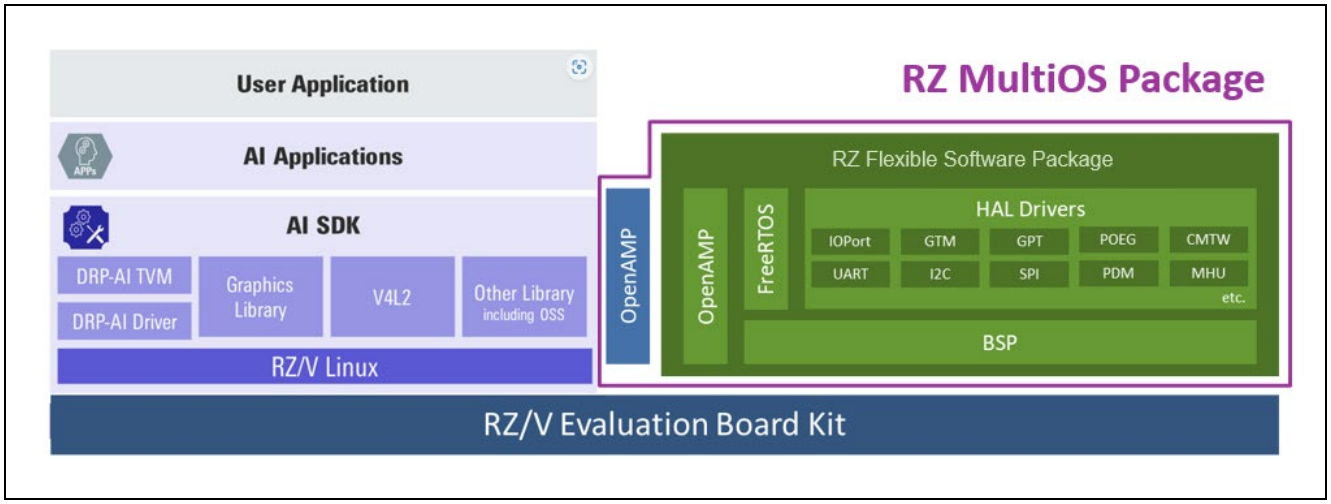
RZ/V2H

Quick Start Guide for RZ Multi-OS Package

Introduction

This document outlines the procedure for integrating the RZ Multi-OS Package into RZ/V2H AI SDK. 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 and/or Cortex-R8 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/V2H:



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/V2H

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.

Table 2-1 Verified Operation Conditions

Item	Contents
Integrated Development Environment	e^2 studio 2025-12 or later
Toolchain	GNU Arm Toolchain 13.3.Rel1 AArch32 bare-metal target (arm-none-eabi)
Dependent Software	• RZ Flexible Software Package (FSP) v4.0.0 • RZ/V2H AI Software Development Kit (SDK) v6.00

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 RZ/V Flexible Software Package.

3.2 Integration of Multi-OS Package related stuff

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

1. Carry out **Step 1**, **Step 2** and **1-8 of Step 3** stated in **How to build RZ/V2H AI SDK Source Code** showcased at AI Applications and AI SDK of RZ/V series.
2. Download Multi-OS Package (r01an8260ej0400-rz-multi-os-pkg.zip) to a working directory and run the commands stated below:

```
$ cd ${YOCTO_WORK}
$ unzip <Multi-OS Dir>/r01an8260ej0400-rz-multi-os-pkg.zip
$ tar zxvf r01an8260ej0400-rz-multi-os-pkg/meta-rz-features_multi-os_v4.0.0.tar.gz
```

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

3. Add the layer for Multi-OS Package.

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

(Optional for remoteproc support)

4. Uncomment the following lines stated in **meta-rz-features/meta-rz-multi-os/meta-rzv2h/conf/layer.conf** for enabling remoteproc support:

```
#MACHINE_FEATURES_append = " RZV2H_CM33_BOOT"
MACHINE_FEATURES_append = " SRAM_REGION_ACCESS"
#MACHINE_FEATURES_append = " CM33_FIRMWARE_LOAD"
#MACHINE_FEATURES_append = " CA55_CPU_CLOCKUP"
MACHINE_FEATURES_append = " CM33_REMOTEPROC"
```

   Be sure NOT to uncomment the above-mentioned 1st, 3rd and 4th line when remoteproc support is enabled.

(Optional for CM33 and CR8 invocation from u-boot)

5. Uncomment the following 2nd line stated in **meta-rz-features/meta-rz-multi-os/meta-rzv2h/conf/layer.conf**. For details on CM33 and CR8 invocation from u-boot, please see 4.4.2.

```
#MACHINE_FEATURES_append = " RZV2H_CM33_BOOT"
MACHINE_FEATURES_append = " SRAM_REGION_ACCESS"
#MACHINE_FEATURES_append = " CM33_FIRMWARE_LOAD"
#MACHINE_FEATURES_append = " CA55_CPU_CLOCKUP"
#MACHINE_FEATURES_append = " CM33_REMOTEPROC"
```

6. Continue to carry out the remaining procedures stated in **Step 3** of **How to build RZ/V2H AI SDK Source Code**.

   Note: The AI SDK includes a setting that disables network source fetching (0000-AI_SDK-settings.patch). For the Multi-OS Package, network access is required to fetch the OpenAMP sources. Please update the following entry in **conf/local.conf**:

   BB_NO_NETWORK = "0"

3.3 Note for integration

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

- 0002-Updated-GTM-channels-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 as shown below:

```
diff --git a/drivers/clk/renesas/r9a09g057-cpg.c b/drivers/clk/renesas/r9a09g057-cpg.c
index 8535afb0d7cd..dff5458deb43 100644
--- a/drivers/clk/renesas/r9a09g057-cpg.c
+++ b/drivers/clk/renesas/r9a09g057-cpg.c
@@ -336,21 +336,21 @@ static const struct rzv2h_mod_clk r9a09g057_mod_clks[] __initconst = {
                  BUS_MSTOP(6, BIT(1))),
   DEF_MOD("rcpu_cmtw3_clkm",    CLK_PLLCLN_DIV32, 4, 2, 2, 2,
                  BUS_MSTOP(6, BIT(2))),
- DEF_MOD("gtm_0_pclk",         CLK_PLLCM33_DIV16, 4, 3, 2, 3,
+ DEF_MOD_CRITICAL("gtm_0_pclk",       CLK_PLLCM33_DIV16, 4, 3, 2, 3,
                  BUS_MSTOP(5, BIT(10))),
- DEF_MOD("gtm_1_pclk",         CLK_PLLCM33_DIV16, 4, 4, 2, 4,
+ DEF_MOD_CRITICAL("gtm_1_pclk",       CLK_PLLCM33_DIV16, 4, 4, 2, 4,
                  BUS_MSTOP(5, BIT(11))),
   DEF_MOD("gtm_2_pclk",         CLK_PLLCLN_DIV16, 4, 5, 2, 5,
                  BUS_MSTOP(2, BIT(13))),
   DEF_MOD("gtm_3_pclk",         CLK_PLLCLN_DIV16, 4, 6, 2, 6,
                  BUS_MSTOP(2, BIT(14))),
- DEF_MOD("gtm_4_pclk",         CLK_PLLCLN_DIV16, 4, 7, 2, 7,
+ DEF_MOD_CRITICAL("gtm_4_pclk",       CLK_PLLCLN_DIV16, 4, 7, 2, 7,
                  BUS_MSTOP(11, BIT(13))),
- DEF_MOD("gtm_5_pclk",         CLK_PLLCLN_DIV16, 4, 8, 2, 8,
+ DEF_MOD_CRITICAL("gtm_5_pclk",       CLK_PLLCLN_DIV16, 4, 8, 2, 8,
                  BUS_MSTOP(11, BIT(14))),
- DEF_MOD("gtm_6_pclk",         CLK_PLLCLN_DIV16, 4, 9, 2, 9,
+ DEF_MOD_CRITICAL("gtm_6_pclk",       CLK_PLLCLN_DIV16, 4, 9, 2, 9,
                  BUS_MSTOP(11, BIT(15))),
- DEF_MOD("gtm_7_pclk",         CLK_PLLCLN_DIV16, 4, 10, 2, 10,
+ DEF_MOD_CRITICAL("gtm_7_pclk",       CLK_PLLCLN_DIV16, 4, 10, 2, 10,
                  BUS_MSTOP(12, BIT(0))),
   DEF_MOD("wdt_0_clkp",         CLK_PLLCM33_DIV16, 4, 11, 2, 11,
                  BUS_MSTOP(3, BIT(10))),
```

Re placing 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/V2H AI SDK

With respect to the deployment of Linux kernel, device tree and root filesystem for RZ/V2H, please refer to https://renesas-rz.github.io/rzv_ai_sdk/6.00/.

4. Sample Program Invocation on RZ/V2H

4.1 Hardware Setup

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

4.2 CM33/CR8 Sample Program Setup

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

1. Extract **r01an8260ej0400-rz-multi-os-pkg.zip** on your development PC.
2. Extract either of **rzv2h_cm33_rpmsg_linux-rtos_example.zip** or **rzv2h_cr8_rpmsg_linux-rtos_emo.zip** included in **r01an8260ej0400-rz-multi-os-pkg**.
3. Open e^2 studio 2025-01 and click **File** > **Import**.
4. Double-click **General** and select **Existing Projects into Workspace** as shown in Figure 4-1:

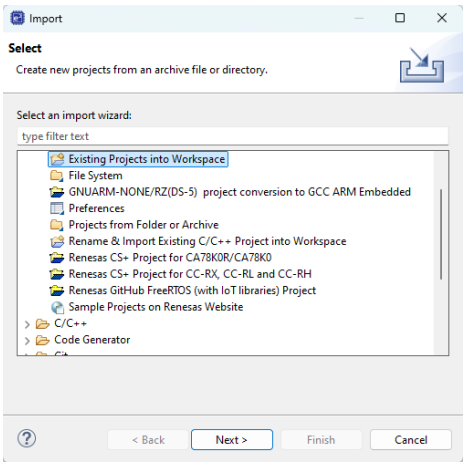


Figure 4-1. Import sample project (1)

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

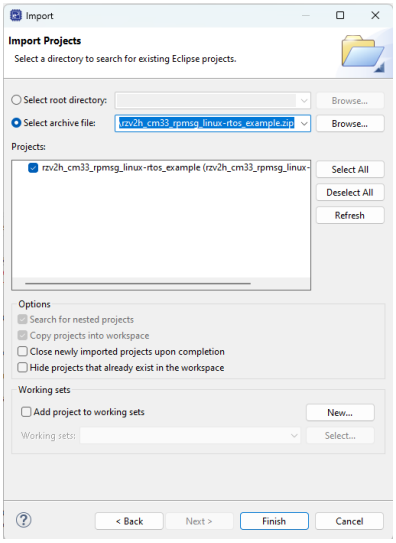


Figure 4-2. Import sample project (2)

(Optional)

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

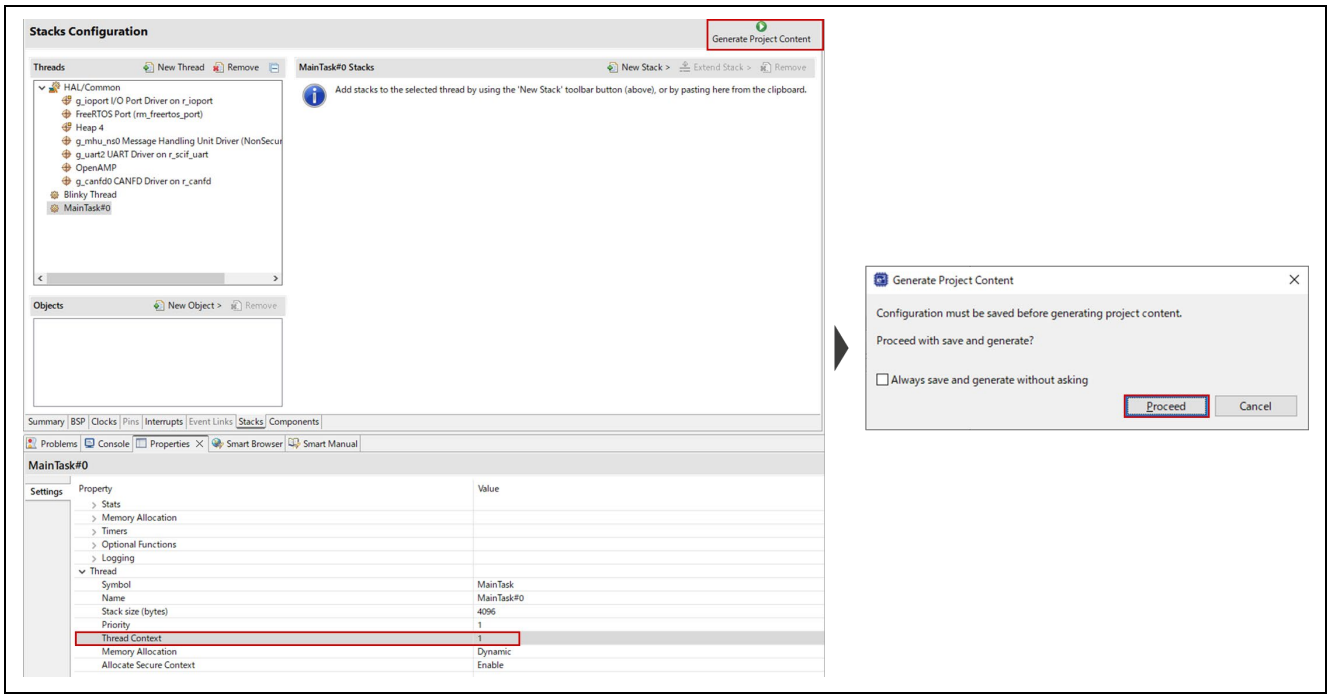


Figure 4-3. RPMsg Channel Setting

7. Build the project from **Choose Project** > **Build Project**.
8. 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.
    - rzv2h_<cpu>_rpmsg_linux-rtos_example.elf
    - rzv2h_<cpu>_rpmsg_linux-rtos_example.bin

    <cpu> stands for either cm33 or cr8 hereafter in this document.

4.3 Note for CR8 Sample Project

- CR8 sample project expects that CM33 and CR8 sample project are imported to the same e^2 studio workspace. Otherwise, FSP Smart Configurator won't work for CR8 sample project.

4.4 CM33/CR8 Sample Program Invocation for communicating with Linux

4.4.1 CM33/CR8 Sample Program Invocation using Segger J-Link

Carry out the procedure shown below for invoking CM33 and/or CR8 sample program using J-Link.

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

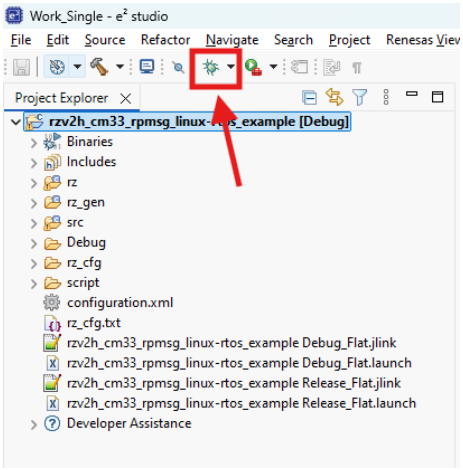


Figure 4-4. Debug Perspective Launch (1)

2. If the following **Select Configuration** window is shown, choose rzv2h_<cpu>_rpmsg_example_Debug_Flat or rzv2h_<cpu>_rpmsg_example_Release_Flat in accordance with the build configuration you are using and click **OK**.

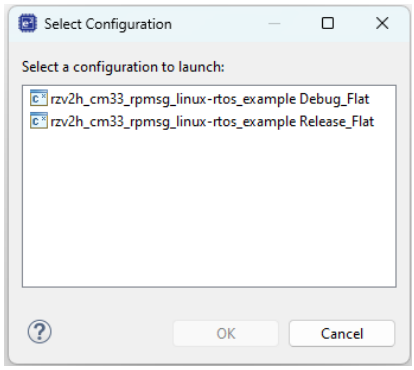


Figure 4-5. Debug Perspective Launch (2)

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

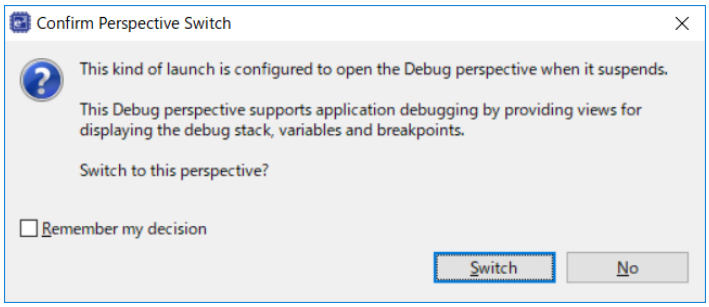


Figure 4-6. Debug Perspective Launch (3)

3. When **Debug Perspective** is opened, Program Counter (PC) should be located at the top of **Entry_Function_S** and **Reset_Handler** for CM33 and CR8 sample project, respectively. Then, press the button shown in Figure 4-7.

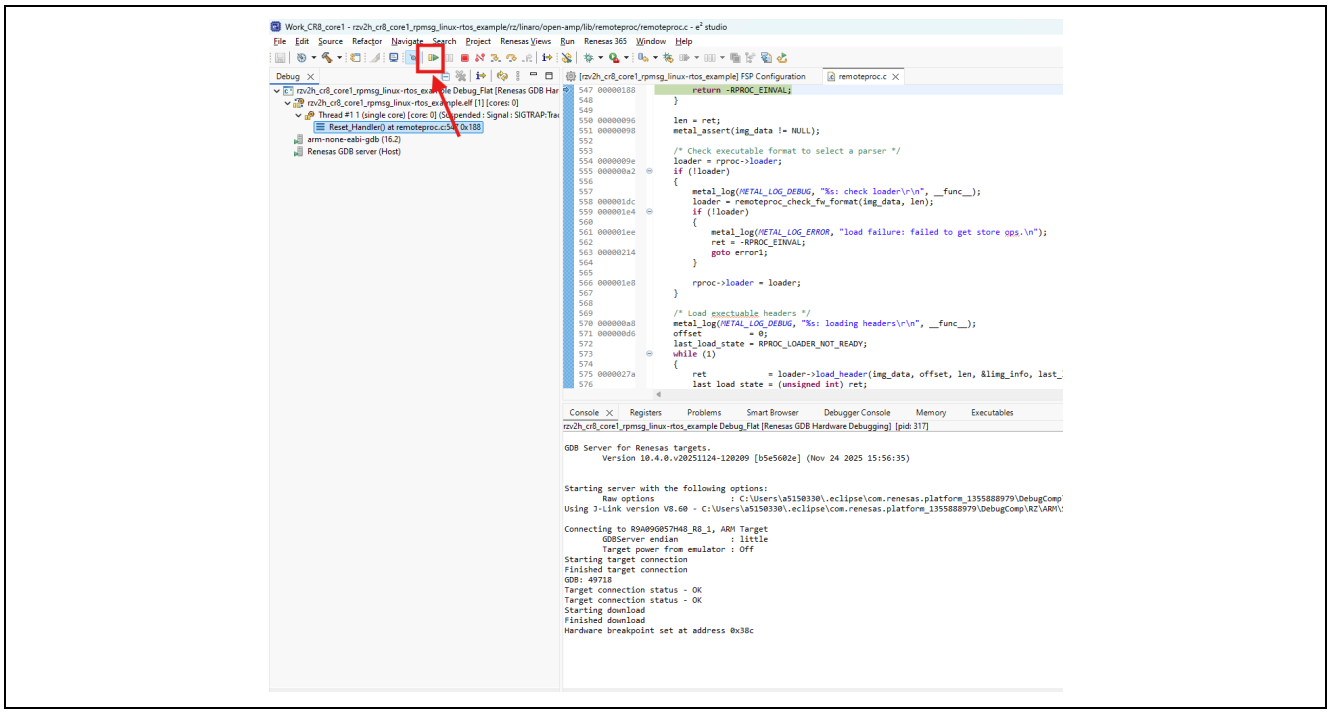


Figure 4-7. How to start to debug Sample Project (1)

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

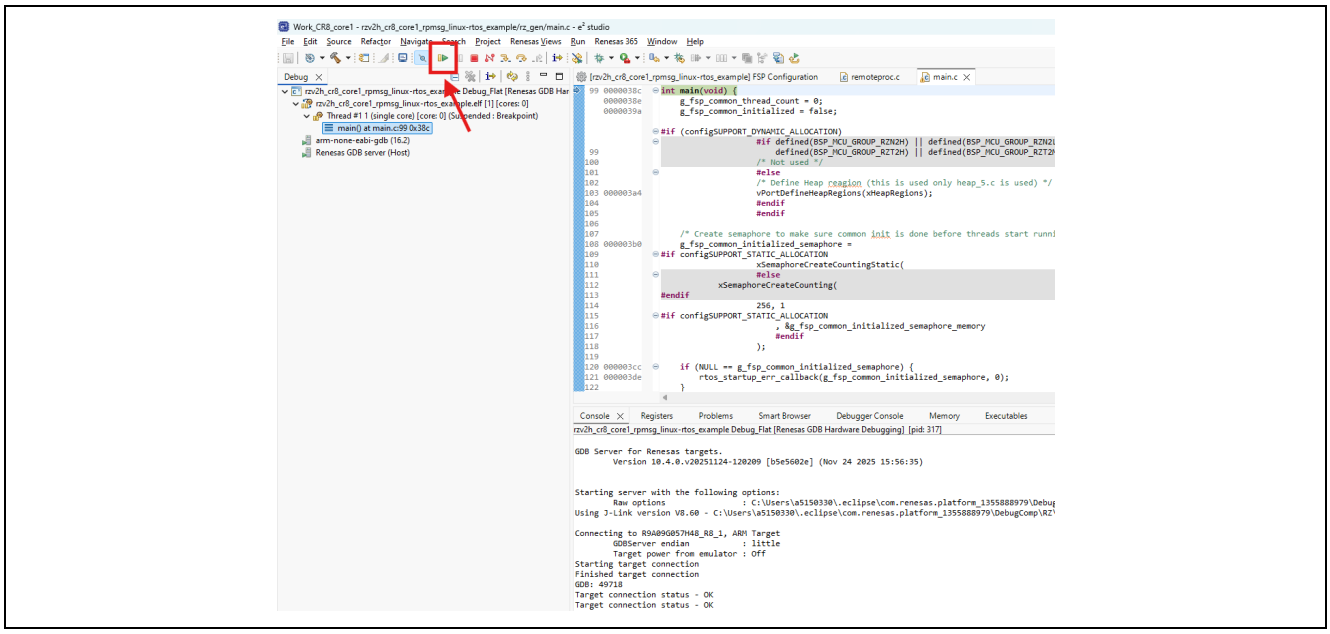


Figure 4-8. How to start to debug Sample Project (2)

5. CM33/CR8 Sample Project starts to work and wait for the launch of CA55 RPMsg Sample Program.

4.4.2 CM33/CR8 Sample Program invocation from u-boot

Here is the example of procedure for invoking CM33/CR8 sample program from u-boot:

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

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

CPU:   Renesas Electronics CPU rev 1.0
Model: Renesas EVK Version 1 based on r9a09g057h4
DRAM:  15.9 GiB
MMC:   mmc@15c00000: 0, mmc@15c10000: 1
(snip)
Net:   eth0: ethernet@15c30000, eth1: ethernet@15c40000
Hit any key to stop autoboot:  3
```

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

- **For CM33**

```
=> setenv cm33start 'dcache off; mw.l 0x10420D2C 0x02000000; mw.l 0x1043080c
0x08003000; mw.l 0x10430810 0x18003000; mw.l 0x10420604 0x00040004; mw.l
0x10420C1C 0x00003100; mw.l 0x10420C0C 0x00000001; mw.l 0x10420904 0x00380008;
mw.l 0x10420904 0x00380038; ext4load mmc 0:2 0x08001e00
boot/rzv2h_cm33_rpmsg_example_linux-rtos.bin; mw.l 0x10420C0C 0x00000000; dcache
on'
=> saveenv
=> run cm33start
```

- **For CR8_Core0**

```
=> setenv cr80start 'dcache off; mw.l 0x10420D24 0x04000000; mw.l 0x10420600
0xE000E000; mw.l 0x10420604 0x00030003; mw.l 0x10420908 0x1FFF0000; mw.l 0x10420C44
0x003F0000; mw.l 0x10420C14 0x00000000; mw.l 0x10420908 0x10001000; mw.l 0x10420C48
0x00000020; mw.l 0x10420908 0x1FFF1FFF; mw.l 0x10420C48 0x00000000; ext4load mmc 0:2
0x12040000 boot/rzv2h_cr8_core0_rpmsg_linux-rtos_example_itcm.bin; ext4load mmc 0:2
0x08180000 boot/rzv2h_cr8_core0_rpmsg_linux-rtos_example_sram.bin; ext4load mmc 0:2
0x40800000 boot/rzv2h_cr8_core0_rpmsg_linux-rtos_example_sdram.bin; mw.l 0x10420C14
0x00000003; dcache on;'
=> saveenv
=> run cr80start
```

- **For CR8_Core1**

```
=> setenv cr81start 'dcache off; mw.l 0x10420D24 0x04000000; mw.l 0x10420600
0xE000E000; mw.l 0x10420604 0x00030003; mw.l 0x10420908 0x1FFF0000; mw.l 0x10420C44
0x003F0000; mw.l 0x10420C14 0x00000000; mw.l 0x10420908 0x10001000; mw.l 0x10420C48
0x00000020; mw.l 0x10420908 0x1FFF1FFF; mw.l 0x10420C48 0x00000000; ext4load mmc 0:2
0x12080000 boot/rzv2h_cr8_core1_rpmsg_linux-rtos_example_itcm.bin; ext4load mmc 0:2
0x081C0000 boot/rzv2h_cr8_core1_rpmsg_linux-rtos_example_sram.bin; ext4load mmc 0:2
0x41800000 boot/rzv2h_cr8_core1_rpmsg_linux-rtos_example_sdram.bin; mw.l 0x10420C14
0x00000003; dcache on;'
=> saveenv
=> run cr81start
```

4.4.3 CM33 Sample Program Invocation from remoteproc

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

1. Booting up Linux by following RZ/V2H EVK Getting Started.
2. Invoke the command stated below to specify the sample program to be loaded:

```
root@rzv2h-evk-ver1:~# echo rzv2h_cm33_rpmsg_linux-rtos_example.elf >
/sys/class/remoteproc/remoteproc0/firmware
```

3. Kick CM33 by invoking the command below:

```
root@rzv2h-evk-ver1:~# 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@rzv2h-evk-ver1:~# echo start > /sys/class/remoteproc/remoteproc0/state
[ 737.289773] remoteprocremoteproc0: powering up cm33
[ 737.348226] remoteprocremoteproc0: Booting fwimage rzv2h_cm33_rpmsg_linux-rtox_example.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
```

4.5 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**.

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

3. Invoke RPMsg sample program as shown below:

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

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

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

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

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


e. exit

please input
>
```

5. By typing either of **1** to **9** corresponding to the communication you would like to try, the sample program starts to work.
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.bhu-shm closed
```

4.6 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 among CA55, CM33 and CR8 is established.
2. The CA55 sample program starts to send the message to CM33 and/or CR8 with incrementing the message size from the minimum value 17 to the maximum value 488. During the communication, the message as shown below is displayed on your console:

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

3. When CM33 and/or CR8 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/CR8 and CM33/CR8 sends back the echo reply, the message indicating the termination of the communication channel is sent from CA55 to CM33/CR8. 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...
```

4.7 Notes for Linux-RTOS Sample Program

When trying 7. Communication with CM33 ch0 and CR8 core 0 ch1, be sure to invoke CM33 RPMsg sample program first followed by CR8 sample program beforehand. If the sample program works as expected, the communications between CA55 and CM33 and between CA55 and CR8 work concurrently as shown in the log below:

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

(snip)

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

4.8 Notes for RTOS-RTOS Sample Program

The current rtos-rtos program is designed to support communication between CM33 and CR8_Core0. To demonstrate communication between CM33 and CR8_Core1, please go to src/platform_info.h in the cm33 rtos-rtos demo and change the value of the TO_CR8_CORE flag from 0 to 1.

```
#define TO_CR8_CORE             (1)     /* 0: CR8 core0, 1: CR8 core1 */
```

If the sample program works as expected, the communications between CM33 and CR8_CoreX work continuously, and the log will be displayed via RTT Viewer as shown below:

```
00> Successfully probed IPI device
00> Successfully open uio device: 42f00000.rsctbl.
00> Successfully added memory device 42f00000.rsctbl.
00> Successfully open uio device: 43100000.vring-ctl1.
00> Successfully added memory device 43100000.vring-ctl1.
00> Successfully open uio device: 43500000.vring-shm1.
00> Successfully added memory device 43500000.vring-shm1.
00> Initialize remoteproc successfully.
00> creating remoteproc virtio
00> initializing rpmsg shared buffer pool
00> initializing rpmsg vdev
00>  1 - Send data to remote core, retrieve the echo and validate its
integrity ..
00> RPMSG service has created.
00> sending payload number 0 of size 9
00> echo test: sent : 9
00>  received payload number 0 of size 9
00> sending payload number 1 of size 10
00> echo test: sent : 10
00>  received payload number 1 of size 10
00> sending payload number 2 of size 11
00> echo test: sent : 11
00>  received payload number 2 of size 11

(snip)

00> sending payload number 468 of size 477
00> echo test: sent : 477
00>  received payload number 468 of size 477
00> sending payload number 469 of size 478
00> echo test: sent : 478
00>  received payload number 469 of size 478
00> sending payload number 470 of size 479
00> echo test: sent : 479
00>  received payload number 470 of size 479
00> sending payload number 471 of size 480
00> echo test: sent : 480
00>  received payload number 471 of size 480
00> **********************************
00>  Test Results: Error count = 0
00> **********************************
00> Quitting application .. Echo test end
```

5. CA55 1.8GHz configuration support at CA55 cold boot mode

This chapter describes how to configure 1.8GHz as operational frequence 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-rzv2h/conf/layer.conf

```
#MACHINE_FEATURES_append = " RZV2H_CM33_BOOT"
#MACHINE_FEATURES_append = " SRAM_REGION_ACCESS"
MACHINE_FEATURES_append = " CM33_FIRMWARE_LOAD"
MACHINE_FEATURES_append = " CA55_CPU_CLOCKUP"
#MACHINE_FEATURES_append = " CM33_REMOTEPROC"
```

   Be sure NOT to uncomment the above-mentioned 1st and 5th line when CA55 1.8GHz support is enabled.

2. Rebuild TrustedFirmware-A as shown below:

```
MACHINE=rzv2h-evk-ver1 bitbake trusted-firmware-a -c cleansstate
MACHINE=rzv2h-evk-ver1 bitbake firmware-pack -c cleansstate
MACHINE=rzv2h-evk-ver1 bitbake core-image-weston
```

3. Deploy build artifacts to SD card by following Renesas RZ/V AI | The best solution for starting your AI applications.

5.2 Setup of CM33 related stuff

1. Import or create RZ/V FSP project for CM33.
2. Open **configurator.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.

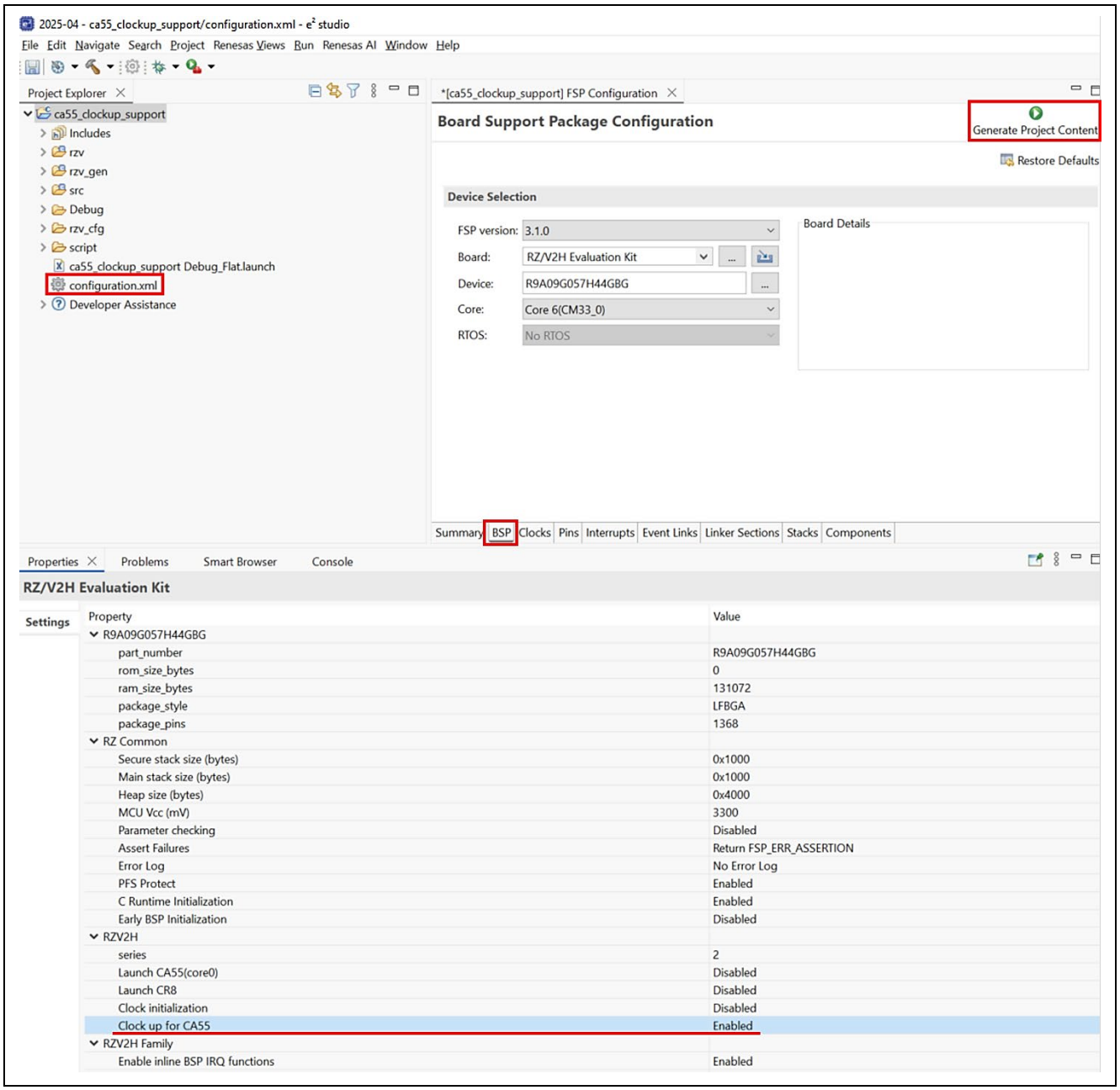


Figure 5-1. CM33 project setting for CA55 1.8GHz support (CA55 coldboot)

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

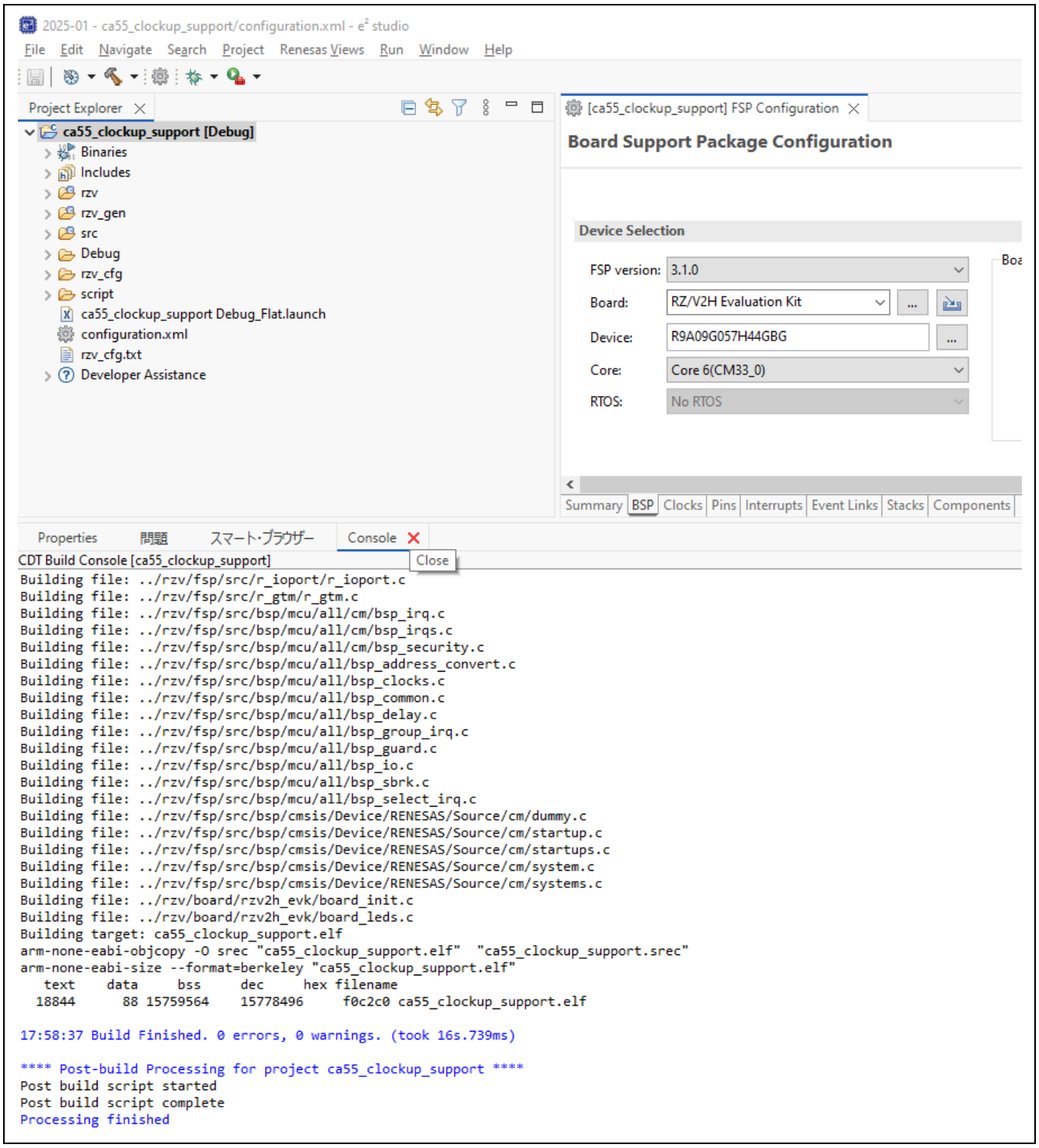


Figure 5-2. Build CM33 project for CA55 1.8GHz support

5.3 Deployment of Build Artifacts to RZ/V2H EVK

This section describes the procedure to program the build artifacts to RZ/V2H EVK.

1. Connect CN12 of RZ/V2H EVK with Host PC and established serial port connection.
2. Configure DSW1-4 and DSW1-5 of RZ/V2H EVK as OFF and ON respectively to specify boot mode as SCIF download mode.
3. Turn on RZ/V2H EVK. Then, the following message is shown on your terminal:

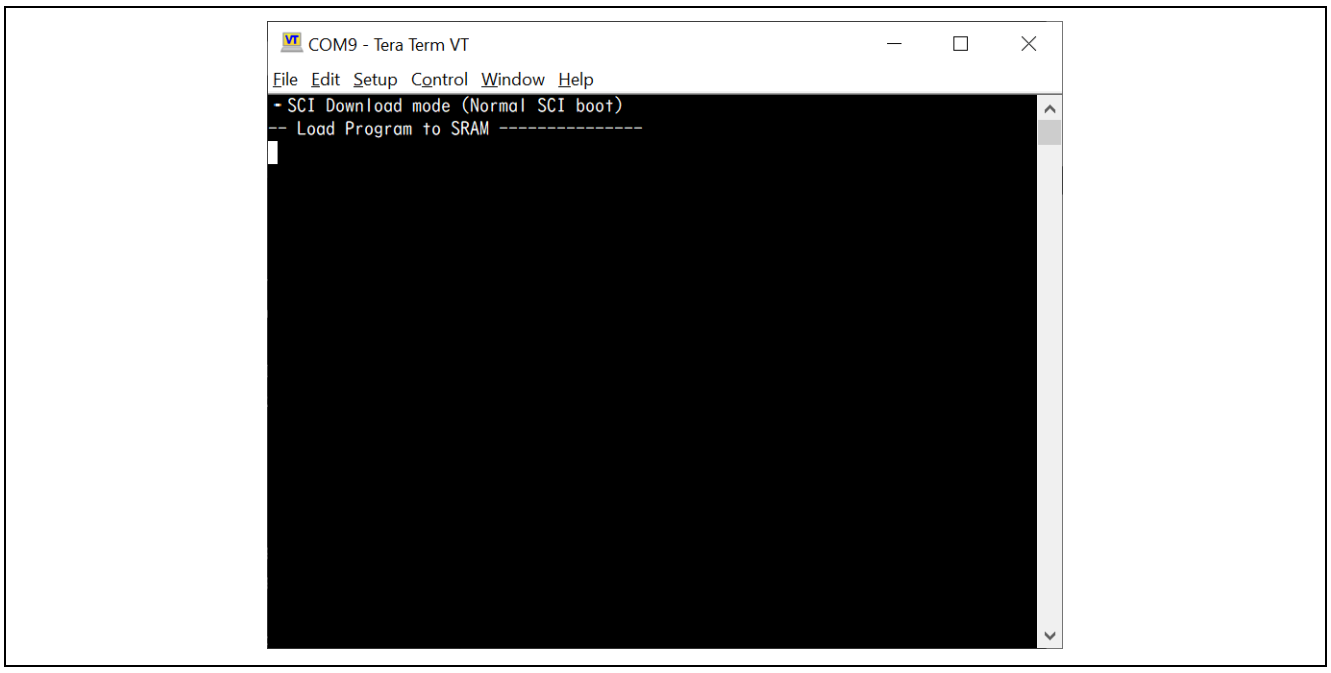


Figure 5-3. SCIF Download mode

4. Send **Flash_Writer_SCIF_RZV2H_DEV_INTERNAL_MEMORY.mot** to RZ/V2H EVK via terminal software. If it's successfully transferred, the following message is shown on your terminal:

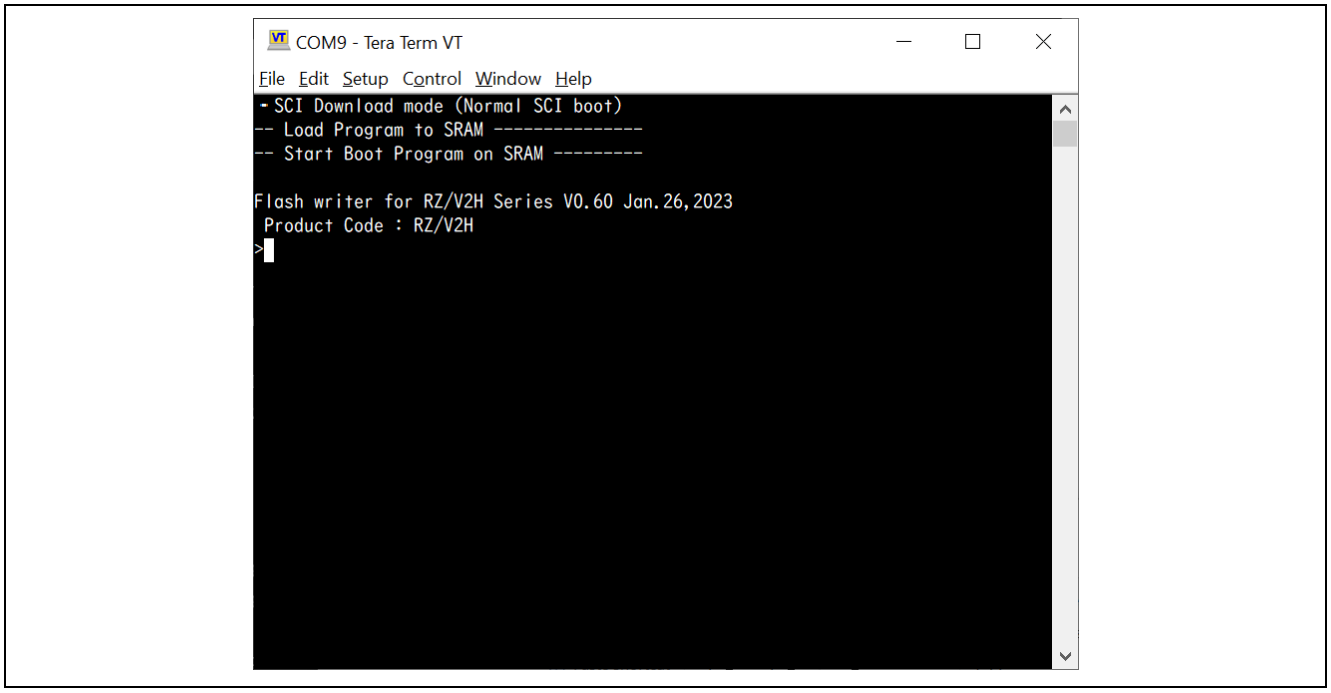


Figure 5-4. Flash Writer invocation

5. Program **bl2_bp_spi-rzv2h-evk-ver1.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'8101E00

===== 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-rzv2h-evk-ver1.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 line in **meta-rz-features/meta-rz-multi-os/meta-rzv2h/conf/layer.conf**.

```
MACHINE_FEATURES_append = " RZV2H_CM33_BOOT"
#MACHINE_FEATURES_append = " SRAM_REGION_ACCESS"
#MACHINE_FEATURES_append = " CM33_FIRMWARE_LOAD"
#MACHINE_FEATURES_append = " CA55_CPU_CLOCKUP"
#MACHINE_FEATURES_append = " CM33_REMOTEPROC"
```

   Be sure NOT to uncomment the above-mentioned 3rd and 4th and 5th line when CM33 cold boot support is enabled.

2. Rebuild TrustedFirmware-A as shown below:

```
MACHINE=rzv2h-evk-ver1 bitbake trusted-firmware-a -c cleansstate
MACHINE=rzv2h-evk-ver1 bitbake firmware-pack -c cleansstate
MACHINE=rzv2h-evk-ver1 bitbake core-image-weston
```

3. Deploy build artifacts to SD card by following Renesas RZ/V AI | The best solution for starting your AI applications.

6.2 Deployment of CA55 Build Artifacts to RZ/V2H EVK

This section describes how to deploy CA55 Build Artifacts to RZ/V2H EVK. First, please carry out the same procedure as 1 to 4 stated in **5.3 Deployment of Build Artifacts to RZ/V2H EVK**. Then, follow the steps stated below:

1. Program **bl2_bp_spi-rzv2h-evk-ver1.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'8101E00

===== 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'00100000
SpiFlashMemory End Address  : H'00136D17
============================================================
```

2. Program **fip-rzv2h-evk-ver1.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'280000
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

1. Import or create RZ/V FSP project for CM33.
2. Open **configurator.xml** in the project and choose **BSP** tab.
3. Configure **Launch CA55(core0)** as Enabled. Also, enabled **Clock up for CA55** if you would like to configure operational frequency of CA55 as 1.8GHz.
4. Click **Generate Project Content** to reflect the changes to your project.

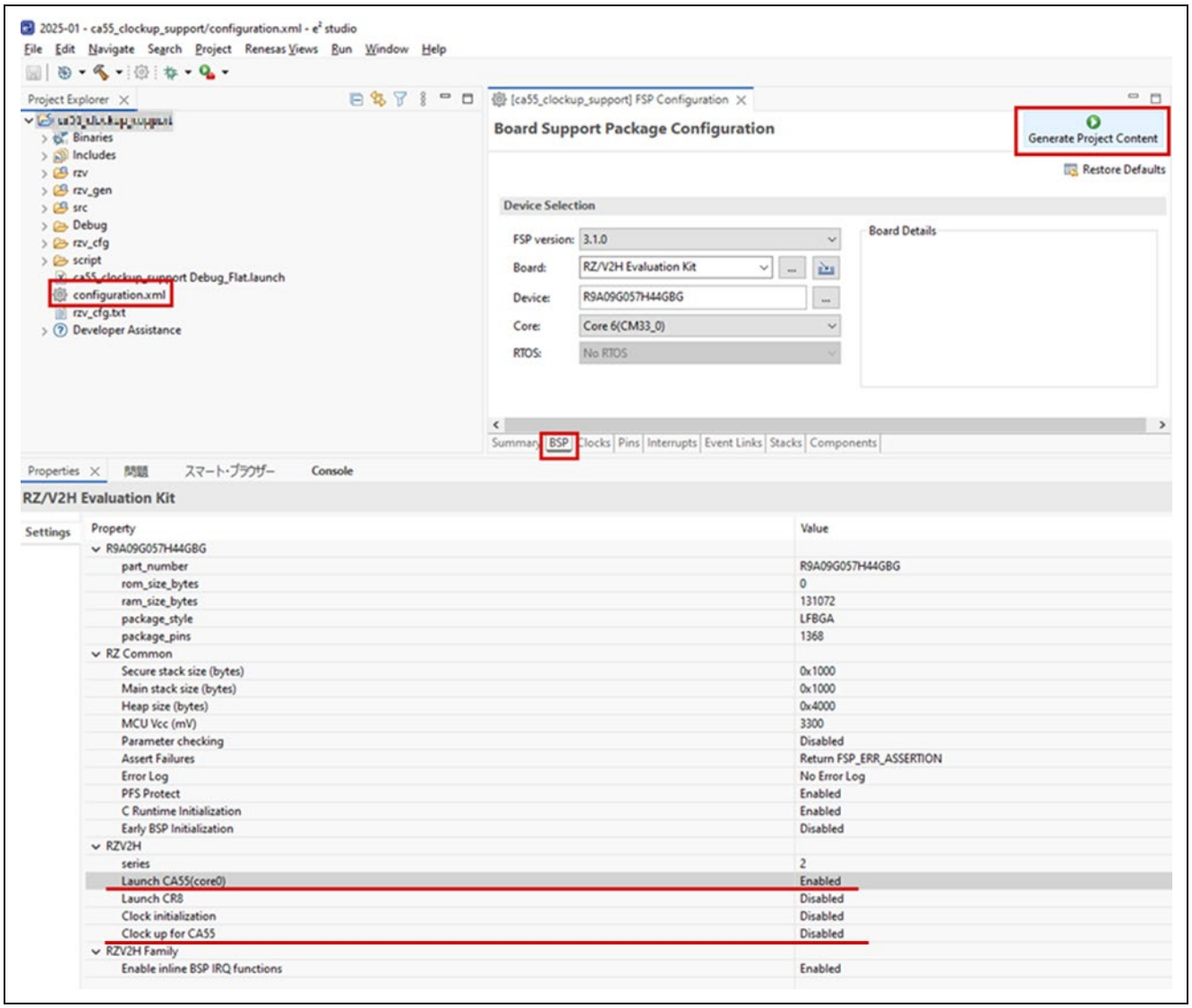


Figure 6-1. CM33 project setting for CM33 cold boot

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

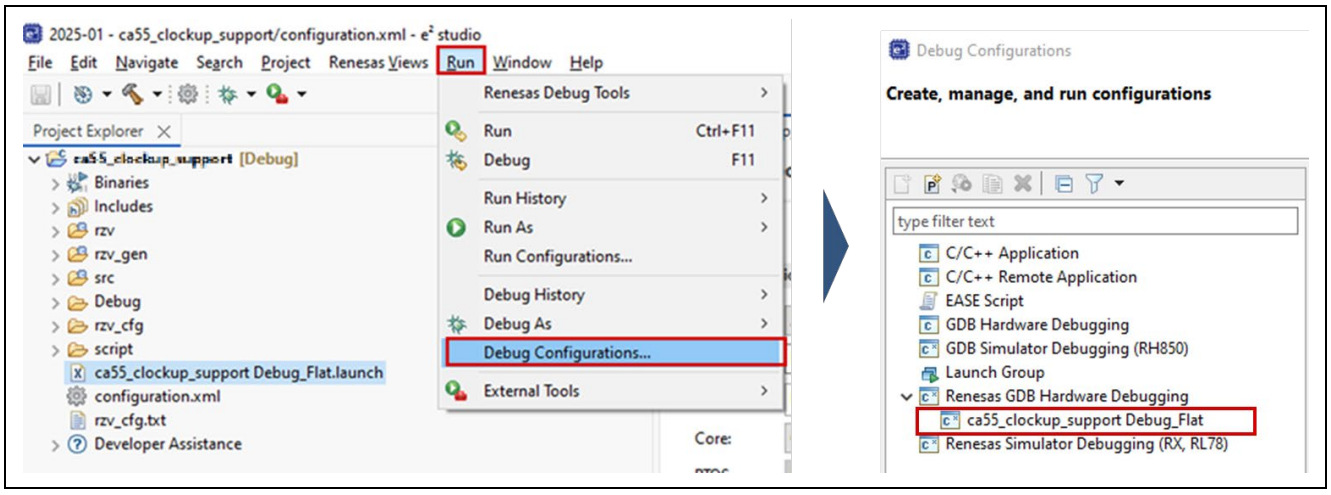


Figure 6-2. Debug Configuration Launch

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

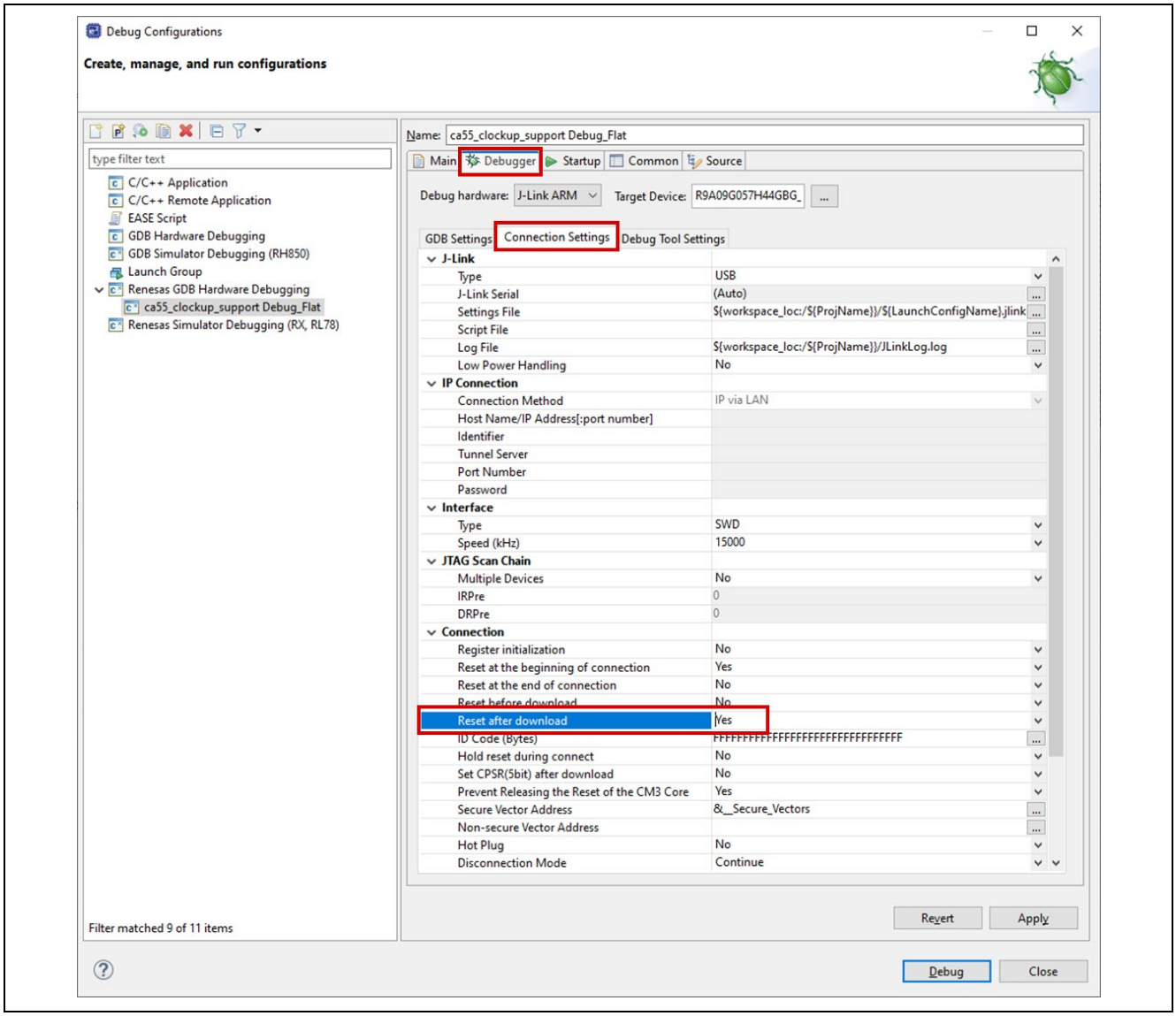


Figure 6-3. Connection Settings

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

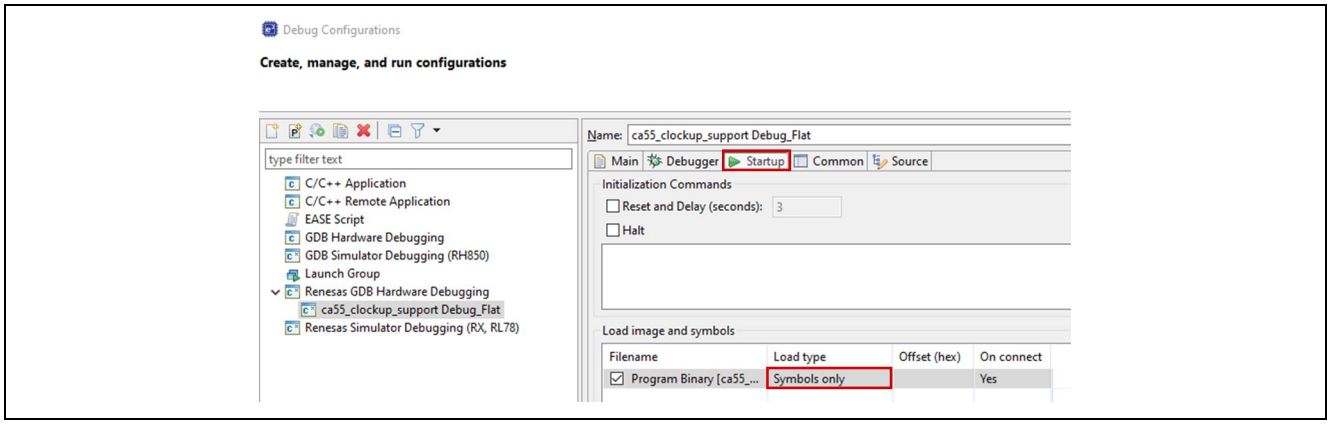


Figure 6-4. Startup Settings (1)

9. Click **Add...** > **Workspace…**, choose **<project name>.srec** and click **OK**.

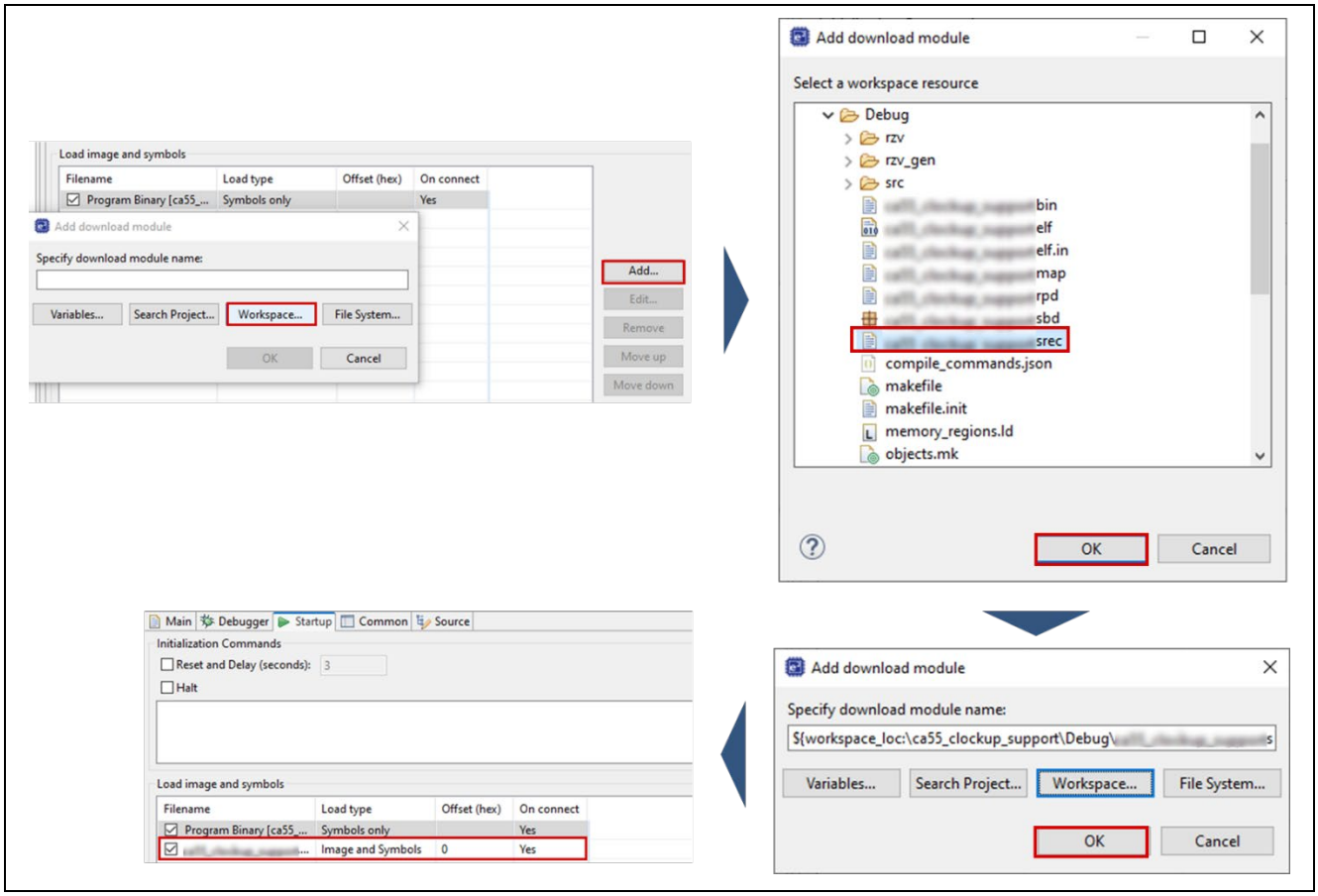


Figure 6-5. Startup Settings (2)

10. Click **Debug** to launch Debug Perspective. Note that DSW1-1, 4, 5 and 7 of RZ/V2H EVK should be specified as OFF, OFF, OFF and ON, respectively before the launch. Then, CM33 program starts, loads CA55 build artifacts from QSPI Flash ROM and kick CA55.

Important Note:
When RZ FSP V4.0.0 is installed, J-Link DLL version 8.60 is installed. This J-Link software is used when e² studio connects to the target board. However, if you are developing software using the CM33 cold boot mode of the RZ/V2H, you MUST use J-Link DLL version 7.96e.
For details on CM33 program invocation with e²studio and how to change from Jlink v8.60 to v7.96e, please refer Getting Started with RZ/V Flexible Software Package.

7. Remoteproc support

This chapter describes how to invoke the CM33 firmware with remoteproc on Linux.

7.1 Setup of CM33 related stuff

Here are the steps to develop your own project based on the projects that support remoteproc included in this package.

1. Import the following projects as base projects.

   — rzv2h_cm33_rpmsg_linux-rtos_example

2. Change the project name (if you need).
   In Project Explorer, right-click on the project name and select “Rename...”

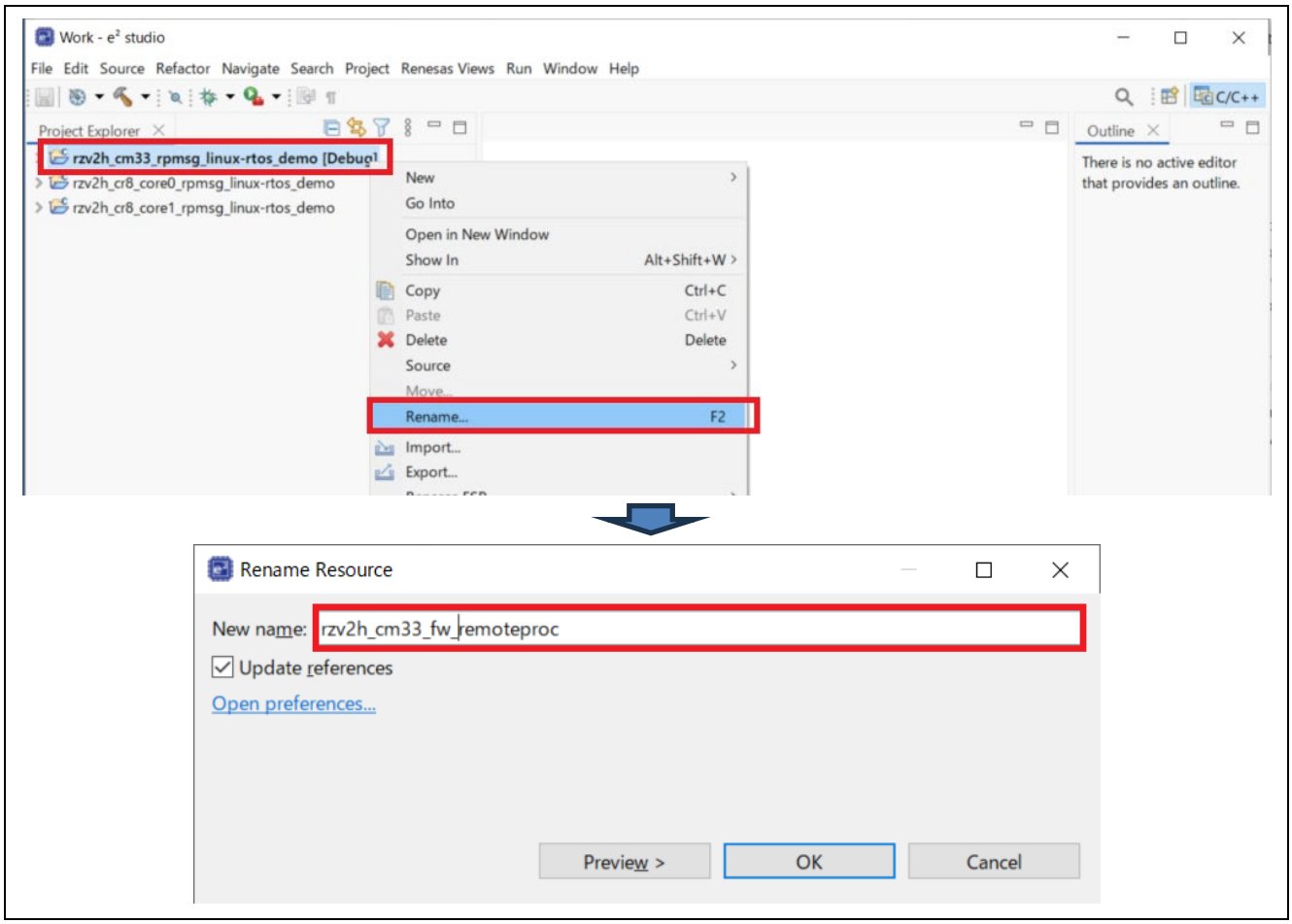


Figure 7-1. Rename project

3. Implement application code.
   Replace the contents of **main_task_entry** function in **src/main_task_entry.c** with the following and implement the application code.

```
/*
* Copyright (c) 2020 - 2024 Renesas Electronics Corporation and/or its affiliates
*
* SPDX-License-Identifier: BSD-3-Clause
*/

#include "main_task.h"
/* Main Task entry function */
/* pvParameters contains TaskHandle_t */
#include "openamp/open_amp.h"
#include "platform_info.h"
#include "rsc_table.h"

extern int app (struct rpmsg_device * rdev, void * platform, unsigned long svcno);

void main_task_entry(void *pvParameters)
{
   unsigned long        proc_id = *((unsigned long*)pvParameters);
   unsigned long        rsc_id  = *((unsigned long*)pvParameters);
   struct rpmsg_device * rpdev;
   void                 * platform;
   int ret;

   ret = init_system();
   if (ret)
   {
      LPERROR("Failed to init remoteproc device.\n");
      goto err1;
   }

   ret = platform_init(proc_id, rsc_id, &platform);
   if (ret)
   {
      LPERROR("Failed to create remoteproc device.\n");
      goto err1;
   }

   // Add your code here

err1:
   vTaskDelete(NULL);

}
```

Figure 7-2. Implement application code.

4. Enable remoteproc.
   Change the value of the **ENABLE_REMOTEPROC** macro in **src/platform_info.h** to **1**.

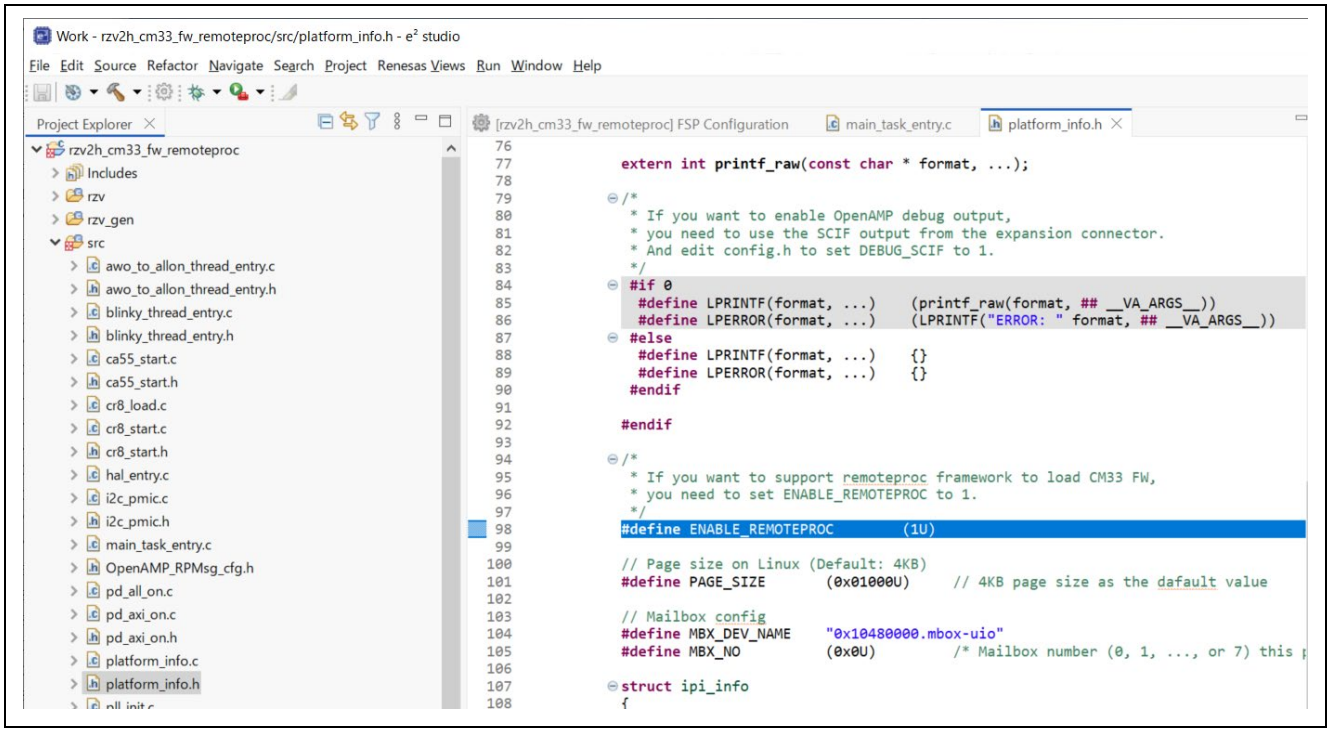


Figure 7-3. Enable remoteproc.

5. Build the project

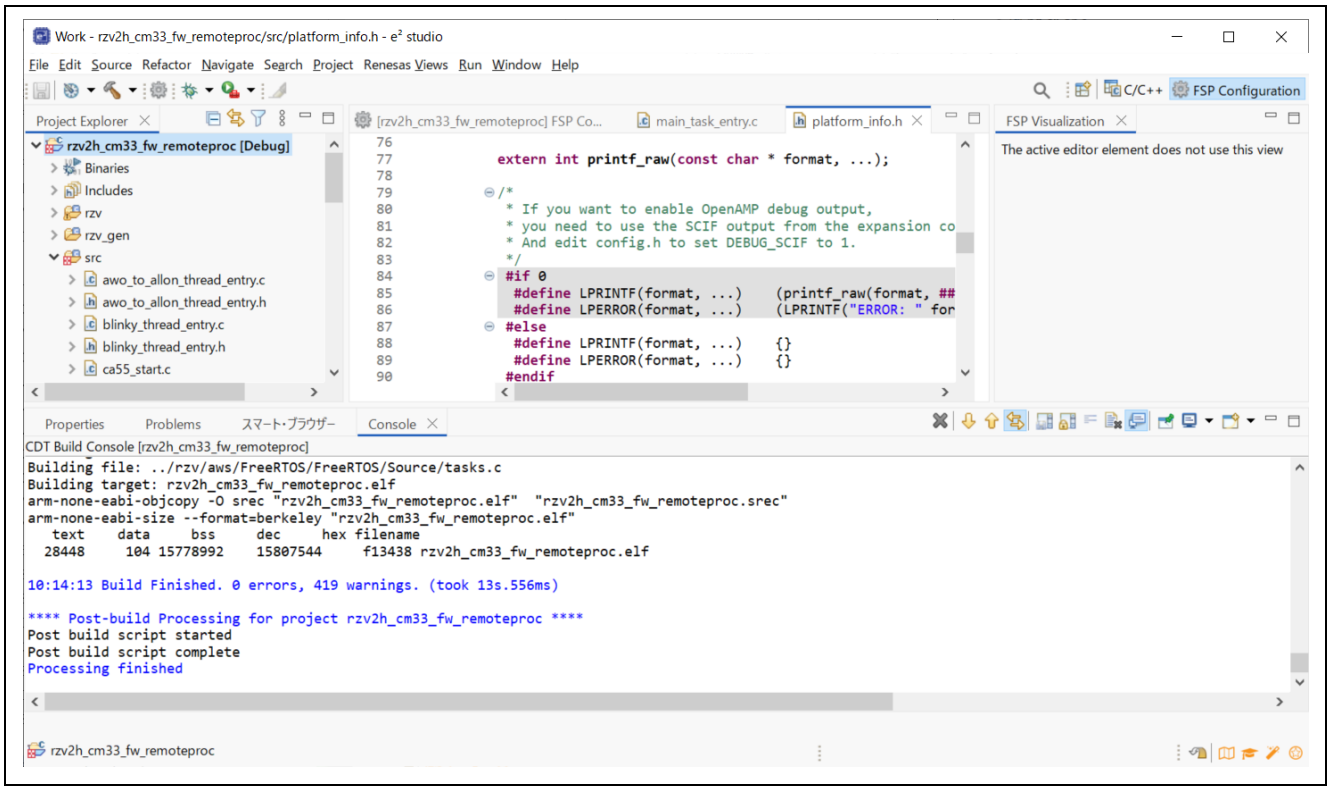


Figure 7-4. Build the project.

6. Copy the image to rootfs.
   Copy the elf file you created in the previous step to the **/lib/firmware** folder on the SD card that contains the Linux rootfs you created in **3.4 Deployment of RZ/V2H AI SDK**.

7.2 How to run CM33 firmware from remoteproc

Follow the steps in chapter **4.4.3 CM33 Sample Program Invocation from remoteproc**.

However, change the echo argument in step 2 to match the file name of the file you copied in step 6 in the previous chapter.

8. Reference Documents

- R01AN6240 RZ/V2L, RZ/V2H, RZ/V2N Getting Started with Flexible Software Package
- R12UZ0147 RZ/V2H Evaluation Board Kit (Secure type) Hardware Manual

Revision History

Rev.	Date	Description	
		Page	Summary
3.00	Mar.11.2025	-	Extract RZ/V2H related description from Release Note of RZ/V Multi-OS Package v2.1.0.
3.10	May.22.2025	-	Updated to align with RZ/V2H AI SDK v5.2.0.
3.11	Jun.13.2025	-	Update Multi-OS Package version to 3.1.1.
3.20	Aug.29.2025	-	Update Multi-OS Package version to 3.2.0.
3.30	Jan.13.2026	-	Updated to align with RZ/V2H AI SDK v6.0.0.
4.00	Mar.31.2026	-	Update Multi-OS Package version to 4.0.0.

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/.