RZ/N2H

Quick Start Guide for RZ Multi-OS Package

Introduction

This document explains the setup procedures for the multi-OS environment and sample programs for a multiple-core system, including CIP Linux running on Cortex®-A55 and Bare-metal/Free-RTOS running on Cortex®-R52 and Cortex®-A55. It also demonstrates how to implement Inter-Processor Communication between these CPU cores.

This package requires the RZ Flexible Software Package (FSP) for a Bare-metal/Free-RTOS environment. The figure below illustrates the software stack for integrating the RZ Multi-OS Package with the RZ/N2H:

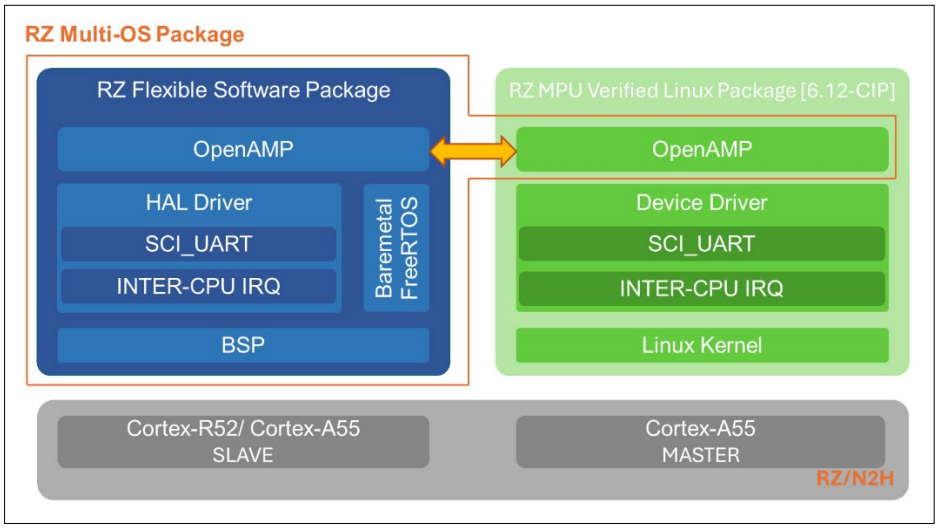


Figure 1-1. Multi-OS Software Architecture with Linux on Cortex®-A55 and Bare-metal / FreeRTOS on Cortex®-R52 and Cortex®-A55

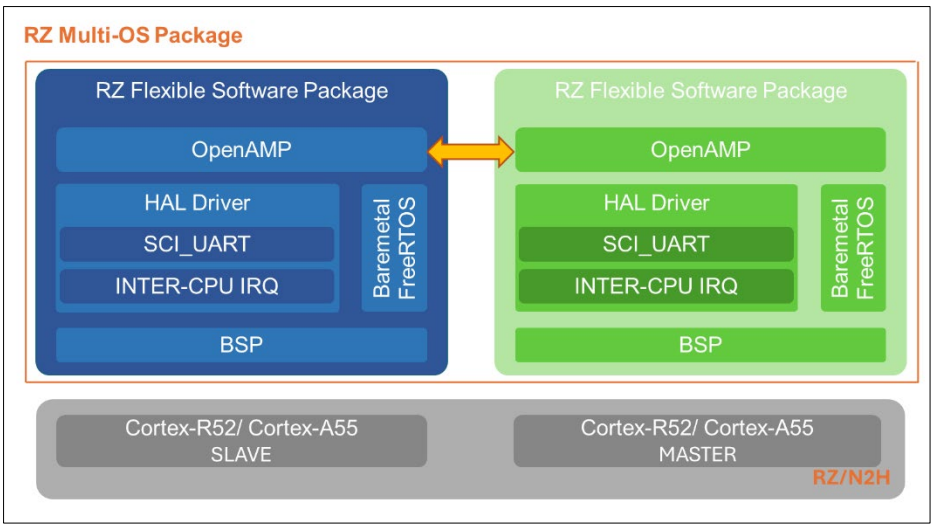


Figure 1-2. Multi-OS Software Architecture with Bare-metal / FreeRTOS on Cortex®-R52 and Cortex®-A55

Here are brief descriptions of each component and concept of the RZ Multi-OS Package for RZ/N2H device:

- RZ FSP
  The software package consists of production-ready peripheral drivers, Bare-metal/Free-RTOS and portable middleware stacks, and the 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.
- FSP - FSP combination
  A combination of two cores in which both cores use the RZ Flexible Software Package.
- Linux - FSP combination
  A combination of two cores in which one core uses Linux (typically CA55_0), and the other cores use the RZ Flexible Software Package.

Target Device

RZ/N2H

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
Serial Communications Interface (SCI)	Performs standard serial communications sending and receiving console messages.
Interrupt controller (ICU)	Configures interrupt settings; the processor will receive interrupts during buffered serial communications; configure inter-processor interrupt.
General Purpose Input Output (GPIO)	Configures I/O lines used by serial communications.

Table 1-2 lists of the equipment to be used in this application.

Table 1-2. Equipment used in this application

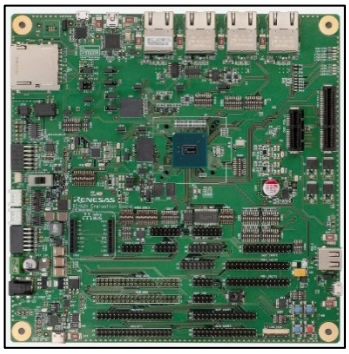
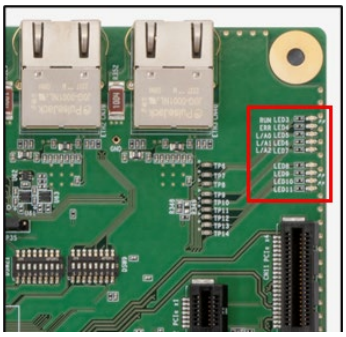
Equipment	Description
Board	RZ/N2H Evaluation Board 
LED	User LED8 on RZ/N2H Evaluation Board 

Table 1-3 lists the mapping structure between the ZIP files and the corresponding platform types.

Table 1-3. Mapping ZIP files to CPU core roles and platform types

CA55				CR52		Project ZIP file
0	1	2	3	0	1	
M	S	-	-	-	-	rzn2h_ca55_0_baremetal_master_ca55_1_baremetal_slave.zip
M	-	S	-	-	-	rzn2h_ca55_0_baremetal_master_ca55_2_baremetal_slave.zip
M	-	-	S	-	-	rzn2h_ca55_0_baremetal_master_ca55_3_baremetal_slave.zip
M	-	-	-	S	-	rzn2h_ca55_0_baremetal_master_cr52_0_baremetal_slave.zip
M	-	-	-	-	S	rzn2h_ca55_0_baremetal_master_cr52_1_baremetal_slave.zip
B	M	S	-	-	-	rzn2h_ca55_1_baremetal_master_ca55_2_baremetal_slave.zip
-	M	-	-	B	S	rzn2h_ca55_1_baremetal_master_cr52_1_baremetal_slave.zip
S	-	-	-	M	-	rzn2h_cr52_0_baremetal_master_ca55_0_baremetal_slave.zip
-	S	-	-	M	-	rzn2h_cr52_0_baremetal_master_ca55_1_baremetal_slave.zip
-	-	S	-	M	-	rzn2h_cr52_0_baremetal_master_ca55_2_baremetal_slave.zip
-	-	-	S	M	-	rzn2h_cr52_0_baremetal_master_ca55_3_baremetal_slave.zip
-	-	-	-	M	S	rzn2h_cr52_0_baremetal_master_cr52_1_baremetal_slave.zip
-	-	-	-	S	M	rzn2h_cr52_1_baremetal_master_cr52_0_baremetal_slave.zip
M	S	-	-	-	-	rzn2h_ca55_0_freertos_master_ca55_1_baremetal_slave.zip
M	-	-	-	S	-	rzn2h_ca55_0_freertos_master_cr52_0_baremetal_slave.zip
B	M	S	-	-	-	rzn2h_ca55_1_freertos_master_ca55_2_baremetal_slave.zip
-	M	-	-	B	S	rzn2h_ca55_1_freertos_master_cr52_1_baremetal_slave.zip
-	-	-	-	M	S	rzn2h_cr52_0_freertos_master_cr52_1_baremetal_slave.zip
-	-	-	-	S	M	rzn2h_cr52_1_freertos_master_cr52_0_baremetal_slave.zip
-	-	-	-	M	S	rzn2h_cr52_0_baremetal_master_cr52_1_freertos_slave.zip
M	-	-	-	S	-	gcc/iar_rzn2h_cr52_0_rpmsg_linux_baremetal_demo.zip
M	-	-	-	-	S	gcc/iar_rzn2h_cr52_1_rpmsg_linux_baremetal_demo.zip
M	S	-	-	-	-	gcc/iar_rzn2h_ca55_1_rpmsg_linux_ baremetal _demo.zip
M	-	S	-	-	-	gcc/iar _rzn2h_ca55_2_rpmsg_linux_ baremetal _demo.zip
M	-	-	S	-	-	gcc/iar_rzn2h_ca55_3_rpmsg_linux_ baremetal _demo.zip
M	-	-	-	S	-	gcc/iar_rzn2h_cr52_0_rpmsg_linux_freertos_demo.zip
M	-	-	-	-	S	gcc/iar_rzn2h_cr52_1_rpmsg_linux_freertos_demo.zip
M	S	-	-	-	-	gcc/iar_rzn2h_ca55_1_rpmsg_linux_freertos_demo.zip
M	-	S	-	-	-	gcc/iar_rzn2h_ca55_2_rpmsg_linux_freertos_demo.zip
M	-	-	S	-	-	gcc/iar_rzn2h_ca55_3_rpmsg_linux_freertos_demo.zip

Note: The symbols and colors in the above table are defined as follows, where M stands for Master project, S for Slave project, and B for Boot project.

Primary core
Baremetal
FreeRTOS
Linux

Table 1-4 lists the boot methods supported in the Linux - FSP example combination.

Table 1-4. Supported boot methods for sample programs

Master	Slave	Debugger	Remoteproc	U-boot	BL2
CA55_0 (Linux)	CA55_1 (Baremetal)	Supported			Supported
CA55_0 (Linux)	CA55_1 (FreeRTOS)	Supported			
CA55_0 (Linux)	CA55_2 (Baremetal)	Supported			Supported
CA55_0 (Linux)	CA55_2 (FreeRTOS)	Supported			
CA55_0 (Linux)	CA55_3 (Baremetal)	Supported			Supported
CA55_0 (Linux)	CA55_3 (FreeRTOS)	Supported			
CA55_0 (Linux)	CR52_0 (Baremetal)	Supported	Supported	Supported	Supported
CA55_0 (Linux)	CR52_0 (FreeRTOS)	Supported	Supported		
CA55_0 (Linux)	CR52_1 (Baremetal)	Supported	Supported	Supported	Supported
CA55_0 (Linux)	CR52_1 (FreeRTOS)	Supported	Supported		

2. Verified operation conditions

Table 2-1 shows the verified operation conditions.

Table 2-1. Verified operating conditions

Item	Contents	Remarks
Integrated Development Environment	e^2 studio 2025-12 or later	Download links: RZ FSP v4.0.0 Refer to the Getting Started document for tool installation.
	RZ Smart Configurator 2025-12 or later	
	IAR Embedded Workbench for ARM 9.60.3	
	SEGGER J-Link 8.60	
Toolchain	CR52: GNU ARM Embedded 13.3.Rel1	
	CA55: GCC ARM A-Profile (Aarch64 bare-metal) 13.2.Rel1	
Dependent Software	RZ Flexible Software Package (FSP) 4.0.0	
	RZ MPU Verified Linux Package 5.0.0	Download links: RZ MPU VLP v5.0.0
	Tera Term 5.2	Download links: Tera Term

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 Multi-OS Package related stuff

This section describes how to integrate OpenAMP related stuff to RZ/N Verified Linux Package (hereinafter referred to as VLP). The steps are based on **RZ/T2H and RZ/N2H Evaluation Board - Linux Start-up Guide** (hereinafter referred to as Linux Start-up Guide) included in RZ MPU VLP V5.0.0.

(1) Follow the procedure stated from the beginning of **2. Build Instruction** of **Linux Start-up Guide**

(2) Download Multi-OS Package (**r01an8260ej0400-rz-multi-os-pkg.zip**) to a working directory and run the commands stated below

```
$ export WORK=~/rz_vlp_v${package version}
$ cd $WORK
$ cp <path to folder download package>/r01an8260ej0400-rz-multi-os-pkg.zip .
$ unzip ./r01an8260ej0400-rz-multi-os-pkg.zip
$ tar zxvf ./r01an8260ej0400-rz-multi-os-pkg/\
meta-rz-features_multi-os_v4.0.0.tar.gz
```

(3) Add the layer for Multi-OS Package, simply run the helper script and select the meta-rzn2h layer from the list

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

Note:

1) To support other optional features of Multi-OS, select the options from (4) to (7) according to your intended use. Please also note that the options from (4) to (7), as well as the configurations described in Section **3.2 Integration of Multi-OS Package related stuff** are only applicable to Linux - FSP combinations.

2) When any of the options from (4) to (7) is selected, after building the VLP, you must re-prepare the Micro-SD card (core-image-minimal-rzn2h-dev.rootfs.wic.gz) and rewrite the bootloader files (bl2_bp_xspi0-rzn2h-dev.srec and fip-rzn2h-dev.srec) to the target board. Details on preparing the Micro-SD card and writing the bootloader files are described in Sections **3. Preparing the SD Card**, **4.6. Download Flash Programmer to RAM**, and **4.7. Write the Bootloader** of the **Linux Start-up Guide**.

(Optional support for the remoteproc sample)

(4) Please modify the macro ENABLE_REMOTEPROC in /meta-rz-multi-os/conf/layer.conf to enable remoteproc support

```
ENABLE_REMOTEPROC ?= "0" : no support for remoteproc (default)
ENABLE_REMOTEPROC ?= "1" : support for remoteproc
```

Note:

When ENABLE_REMOTEPROC ?= “1”, setting RPMSG_REMOTE_CORE ?= “0” is required for the sample program to work.

(Optional support for the U-boot sample)

(5) Please modify the macro ENABLE_U_BOOT in /meta-rz-multi-os/conf/layer.conf to enable u-boot support

```
ENABLE_U_BOOT ?= "0" : no support for u-boot (default)
ENABLE_U_BOOT ?= "1" : support for u-boot
```

Note:

When ENABLE_U_BOOT ?= “1”, setting RPMSG_REMOTE_CORE ?= “0” is required for the sample program to work.

(Optional support for the BL2 of Trusted Firmware-A sample)

(6) Please remove “#” in front of MACHINE_FEATURES_append in /meta-rz-multi-os/conf/layer.conf to enable BL2 of Trusted Firmware-A sample support

We need to enable at least the following description.

```
MACHINE_FEATURES_append = " RZN2H_CA55_CR52_BL2_BOOT"
```

And, select target core.

Ex: When CR52_0 is the slave core.

```
MACHINE_FEATURES_append = " RZN2H_CR520_BL2_BOOT"
```

(Optional support for CA55_1/2/3 FSP)

(7) To support CA55 core for FSP, please modify the macro RPMSG_REMOTE_CORE in /meta-rz-multi-os/conf/layer.conf

```
RPMSG_REMOTE_CORE ?= "0" : sample program for FSP (CR52) - Linux (default)
RPMSG_REMOTE_CORE ?= "1" : sample program for FSP (CA55) - Linux
```

(8) Start a build as described in (5) Start a build of 2.1 Building Images as shown below

```
$ MACHINE=rzn2h-dev bitbake core-image-minimal
```

After completing the build with the Multi-OS Package, refer to Sections **3. Preparing the SD Card**, **4.6. Download Flash Programmer to RAM**, and **4.7. Write the Bootloader** of the **Linux Start-up Guide** to prepare the SD card and flash the bootloader to the board.

4. Sample program invocation

4.1 Overview of application behavior

4.1.1 Overview of application for FSP - FSP

The behavior of the application is as follows:

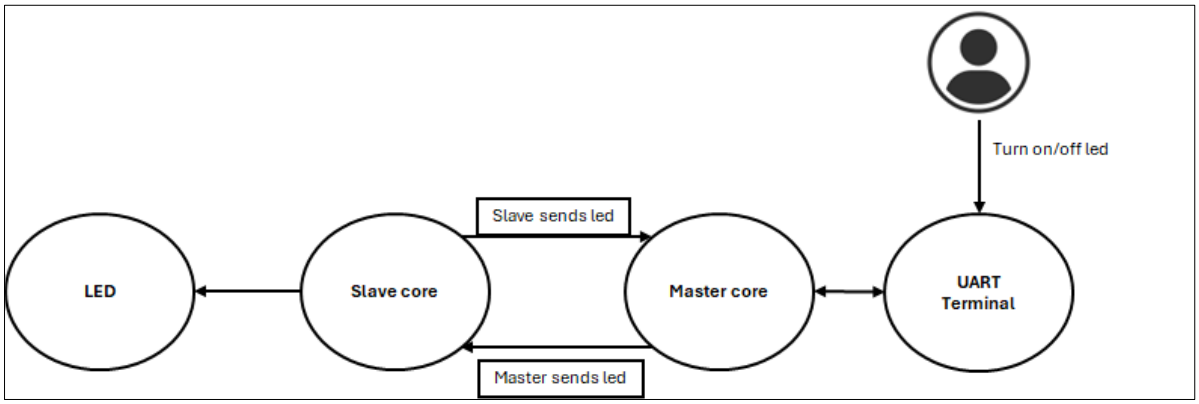


Figure 4-1. The behavior of the application for FSP - FSP

(1) Setting up running for the master core and slave core.

(2) UART console of Master core and LED of Slave core can operate to control LED ON and LED OFF.

(3) Master core reads data from UART that is to turn off/on for led as data below:

Data	Action
43210C01	Turn on LED
43210C00	Turn off LED
Anything	Do nothing

(4) After completing receive data, the master core will send states to turn on/off to the slave core.

(5) The slave core receives a state of led from the master core through OpenAMP. The slave core executes turn on/ off LED. Then, the slave core sends back to inform slave's implementation for led to the master core through OpenAMP.

(6) The master core receives slave's implementation for led through OpenAMP and print to console "Slave turned on LED" "Slave turned off LED".

4.1.2 Overview of application for Linux – FSP

The behavior of sample program is as follows:

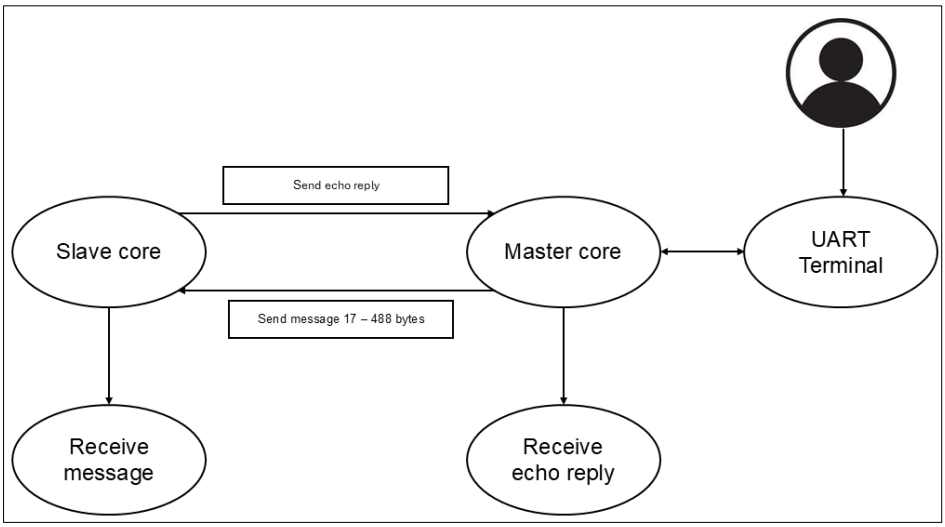


Figure 4-2. The behavior of the application for Linux – FSP

(1) Wait until a communication channel between FSP and Linux is established.

(2) Once the communication channel is established, Linux sample program starts to send the message to FSP while increasing its size from the minimum value 17 to the maximum value 488. At that time, the message like the following should be shown in the console connected to USB to Serial Port of RZ/N2H board.

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

(3) When FSP receives the message sent from Linux, the echo reply is sent back to Linux.

(4) When Linux receives the echo reply, the message below should be displayed in the console connected to USB to Serial Port of RZ/N2H board.

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

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

Termination message on Linux side:

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

Termination message on FSP side:

```
De-initializating remoteproc
```

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

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

Note:

1) Generic Interrupt Controller (GIC) Configuration Handling Between Linux and FSP.

- The initial configuration of the GIC follows the order: Linux first, then FSP. This means any overlapping configurations to the GICD (Distributor) registers made by Linux will be overwritten by the FSP during its initialization.
- To avoid unexpected behavior or interrupt conflicts, make sure the same interrupt ID is not used on both the Linux and FSP sides.
- If an interrupt ID is configured by both Linux and FSP, the FSP's routing configuration will take precedence, potentially altering the expected Linux behavior.

2) U-boot initial settings and ETH3 Port Usage

- The RZ Multi-OS Package modifies the U-Boot initial settings to support Multi-OS configurations. These modifications affect peripheral initialization and default port assignments. Port 33 is reassigned from ETH3 to SCI1 for Multi-OS communication. As a result:
    - Linux boot over the network (e.g., TFTP/NFS via ETH3) cannot be used.
    - ETH3 interface is unavailable while SCI1 is active.

4.2 Hardware setup

4.2.1 Hardware setup when using e² studio

(1) Connect J-Link OB to RZ/N2H

(2) Connect the USB to Serial Port to the RZ/N2H

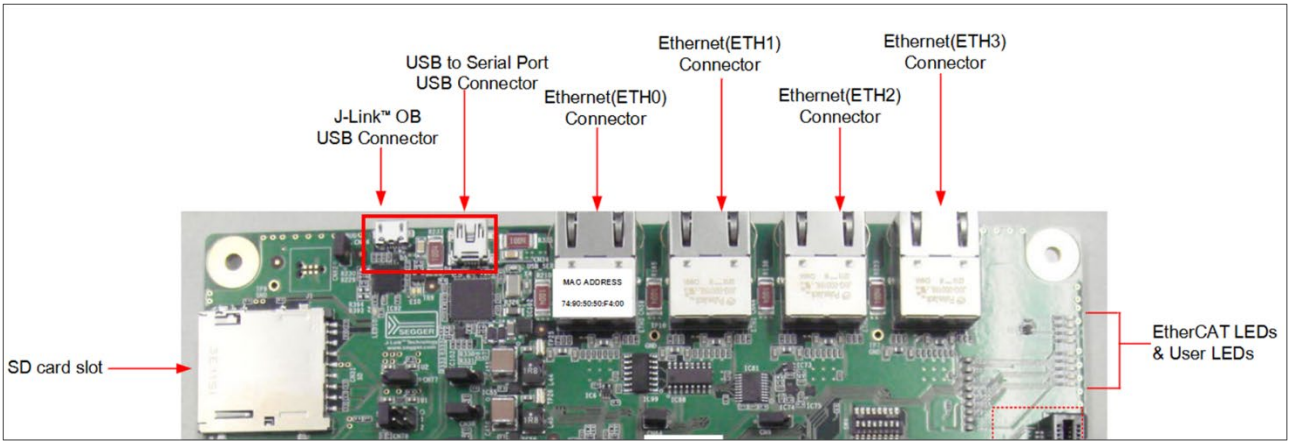


Figure 4-3. Connection J-Link, SCI (UART) to RZ/N2H board

(3) Setup operation mode switch setting for RZ/N2H evaluation board

Table 4-1 lists the required switch configurations when booting FSP - FSP combinations.

Table 4-1. Switch configurations for FSP - FSP combinations

Switch	Setting	Description
DSW3.1	ON	xSPI1 boot mode (x1 boot serial flash)
DSW3.2	OFF	
DSW3.3	ON	
DSW3.4	OFF	CPU0 ATCM wait cycle = 1 wait cycle
DSW3.5	OFF	CPU1 ATCM wait cycle = 1 wait cycle
DSW3.6	OFF	Supply voltage of boot peripheral is 3.3 V
DSW3.7	ON	JTAG mode = Normal mode
DSW15.8	OFF	LED8 (Green)
DSW15.9	OFF	
DSW15.10	ON	
DSW9.1	ON	Reception of UART data 1 for USB-to-serial conversion Transmission of UART data 1 for USB-to-serial conversion
DSW9.2	OFF	
DSW9.3	ON	
DSW9.4	OFF	
DSW5.8	OFF	Reception of UART data 2 for USB-to-serial conversion Transmission of UART data 2 for USB-to-serial conversion
DSW9.5	ON	
DSW9.6	OFF	
DSW9.7	ON	
DSW9.8	OFF	

Table 4-2 lists the required switch configurations when booting Linux - FSP combinations.

Table 4-2. Switch configurations for Linux - FSP combinations

Switch	Setting	Description
DSW3.1	ON	xSPI0 boot mode (x1 boot serial flash)
DSW3.2	ON	
DSW3.3	ON	
DSW3.4	OFF	CPU0 ATCM wait cycle = 1 wait cycle
DSW3.5	OFF	CPU1 ATCM wait cycle = 1 wait cycle
DSW3.6	OFF	Supply voltage of boot peripheral is 3.3 V
DSW3.7	ON	JTAG mode = Normal mode
DSW5-3	ON	Signal Connections of MicroSD Card Slot
DSW19-1	OFF	
DSW19-2	ON	
DSW5-8	OFF	Signal Connections of USB-to-Serial Conversion

4.2.2 Hardware setup when using EWARM

(1) Connect I-Jet to RZ/N2H

(2) Connect USB to serial port to RZ/N2H

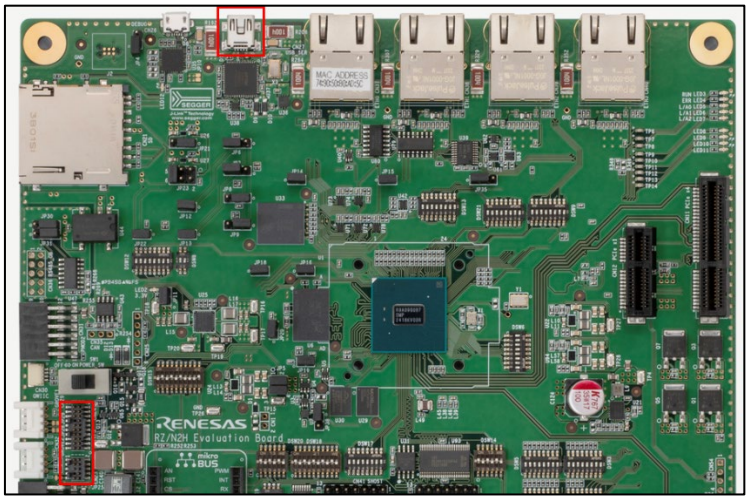


Figure 4-3. Connection J-Link, SCI (UART) to RZ/N2H board

(3) Setup operation mode switch setting for RZ/N2H evaluation board

Refer to **Table 4-1. Switch Configurations for FSP - FSP Combinations** and **Table 4-2. Switch Configurations for Linux - FSP Combinations**.

4.3 Sample program setup on e² studio

Please carry out the following procedures for setting up the demo program running on e² studio.

(1) Extract **r01an8260ej0400-rz-multi-os-pkg.zip** on your development PC

(2) Extract the GCC project

The project for **FSP - FSP** combinations follows the naming format below:

<*device*>_<*platform_type*>_<*role*>_<*device*>_<*platform_type*>_<*role*>.zip

Where:

<device>: rzn2h_cr52_0 / rzn2h_cr52_1 / rzn2h_ca55_0 / rzn2h_ca55_1 / rzn2h_ca55_2 / rzn2h_ca55_3

<platform_type>: baremetal / freertos

<role>: master / slave

(e.g., rzn2h_ca55_0_baremetal_master_ca55_1_baremetal_slave.zip)

Meanwhile, the project for **Linux - FSP** combinations follows this format:

gcc_<*device*>_rpmsg_linux_<*platform_type*>_demo.zip

Where:

<device>: rzn2h_cr52_0 / rzn2h_cr52_1 / rzn2h_ca55_1 / rzn2h_ca55_2 / rzn2h_ca55_3

<platform_type>: baremetal / freertos

<role>: master / slave

(e.g., gcc_rzn2h_ca55_1_rpmsg_linux_baremetal_demo.zip)

(3) Open e² studio and click **File >> Import**

(4) Double-click General and select Existing Projects into Workspace

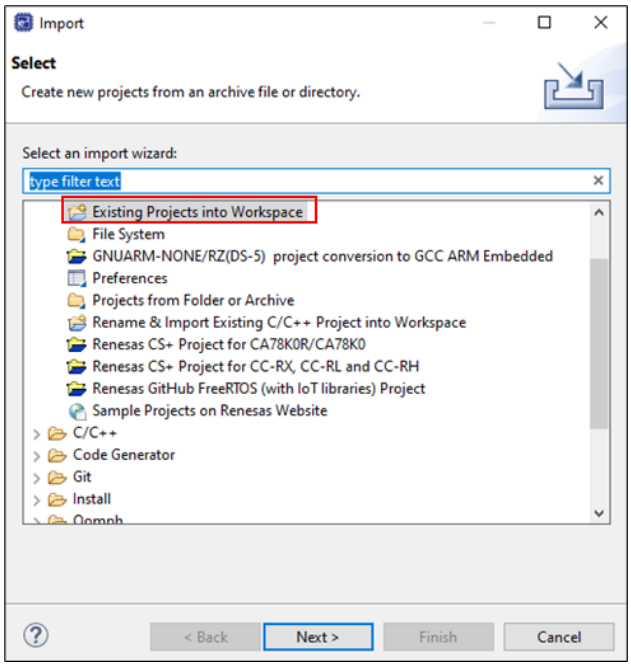


Figure 4-5. Import sample project on e² studio

(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.

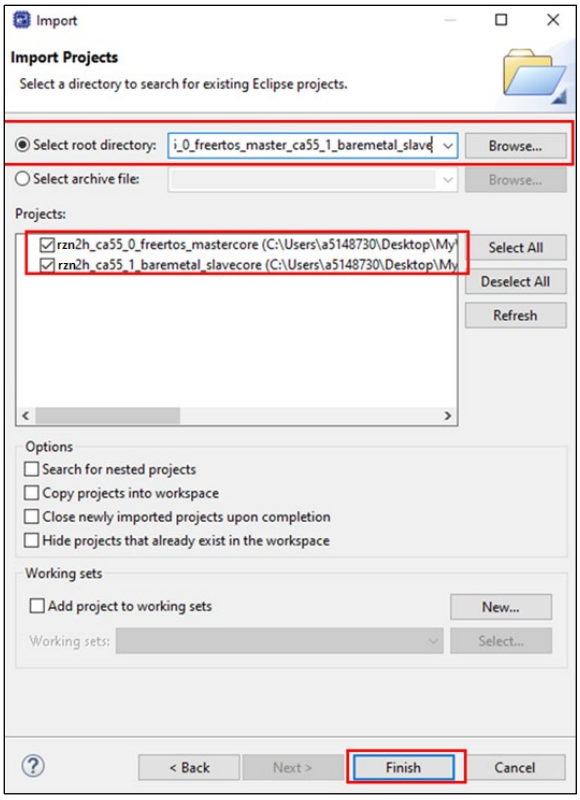


Figure 4-6. Importing the sample project in e2 studio for an FSP - FSP combination

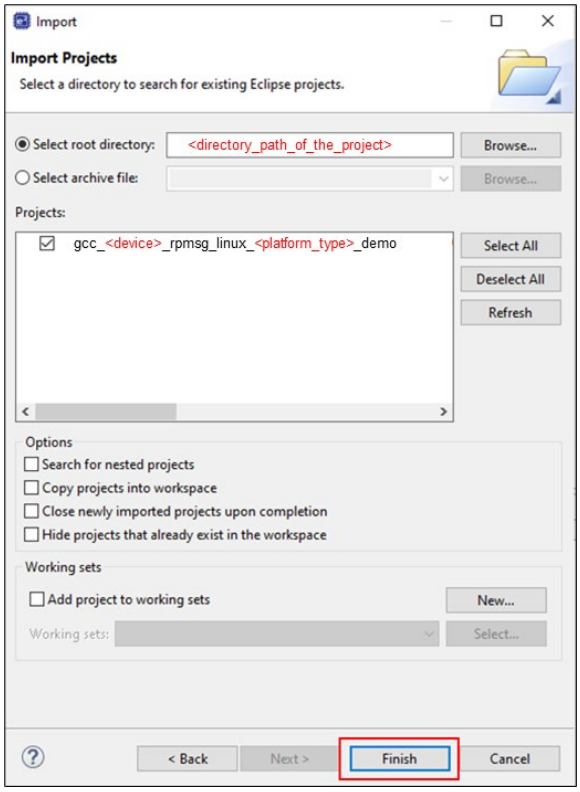


Figure 4-7. Importing the sample project in e2 studio for an Linux - FSP combination

Note:

1) For FSP - FSP combinations, since both cores use FSP, you need to import both projects, including one master and one slave, for the example.

2) For Linux - FSP combinations, since one core uses FSP and one core uses Linux, you only need to import one project for the example.

4.3.1 Setting for sample program to invocation with Segger J-Link

When invoking with Segger J-Link, perform the following step:

(1) Build the project by selecting Choose **Project** >> **Build Project**

Note:

1) For FSP - FSP combinations, the build priority for each core is required.

Table 4-3 lists the build priority order for the FSP - FSP combination example.

Table 4-3. Priority Order When Building the Project

Priority	Project
1 (Highest)	cr52_0 project/ ca55_0 project (primary core)
2	cr52_1 project
3	ca55_0 project (secondary core)
4	ca55_1 project
5	ca55_2 project
6 (Lowest)	ca55_3 project

2) For Linux - FSP combinations, if the example project is not for the primary core, rzn2h_cr52_0_boot must be built first.

(3) For program invocation, refer to Section **4.5.1 Multi-Cores sample program invocation with Segger J-Link on e^2 studio**.

(Optional setting for the remoteproc sample)

4.3.2 Setting for sample program to invocation with remoteproc

When invoking remoteproc on a Linux - FSP combination, perform the following step:

(1) Build the project by selecting Choose Project >> Build Project.

(2) For program invocation, refer to Section **4.5.3 Multi-cores sample program invocation with remoteproc.**

Note:

By default, the firmware for the remoteproc sample is pre-prepared in “/meta-rz-multi-os/recipes-firmware /cr52-firmware/files”. Therefore, rebuilding and reloading the sample is only necessary when modifications are made.

(Optional setting for the u-boot sample)

4.3.3 Setting for sample program to invocation with u-boot

When invoking u-boot on a Linux - FSP combination, perform the following step:

(1) In the Clocks tab configuration, navigate to **CLMA6 Enable** >> **CLMA6 Alternative CLK** and set it to **PLL**.

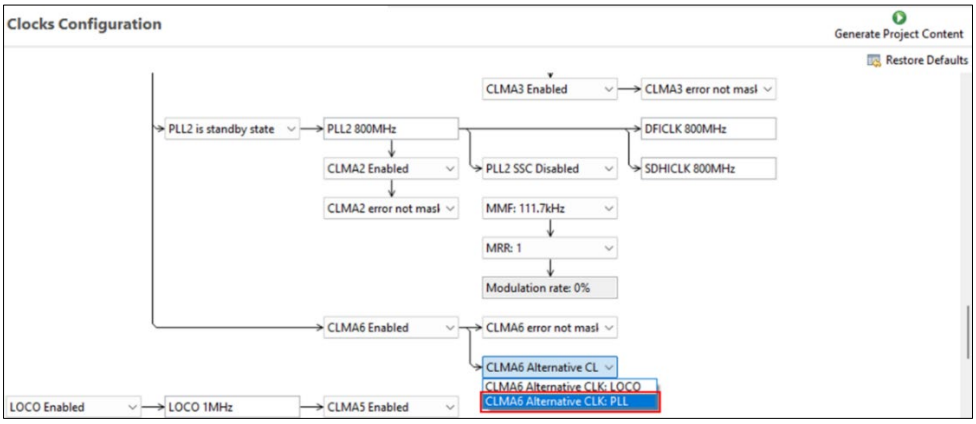


Figure 4-8. Clocks tab configuration

Note:

Please note that this configuration is performed on the primary core (CR52_0).

(2) Click the Generate Project Content button to complete the process.

(3) Navigate to Choose Project >> Properties >> C/C++ Build >> Settings >> Tool Settings >> Cross ARM GNU Create Flash Image >> General, then set Output file format (-O) to Raw binary and remove --gap-fill 0xff from Other flags. Click Apply and Close to complete.

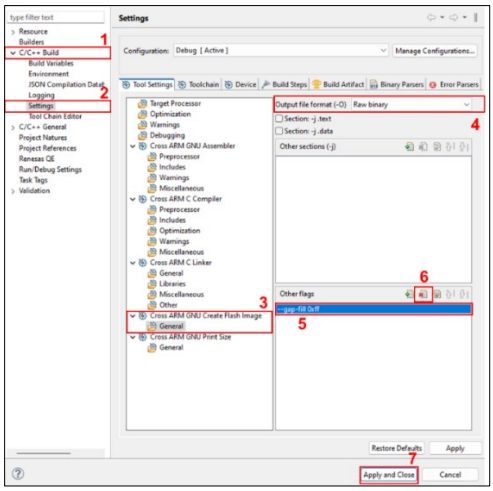


Figure 4-9. Firmware Binary Configuration on e2 Studio

(4) Build the project by selecting Choose Project >> Build Project. The .bin file can be found in the Debug folder.

(5) For program invocation, refer to Section **4.5.4 Multi-cores sample program invocation with u-boot.**

(Optional setting for the BL2 of Trusted Firmware-A sample)

4.3.4 Setting for sample program to invocation with BL2 of Trusted Firmware-A

When invoking BL2 of Trusted Firmware-A on a Linux - FSP combination, perform the following step:

(1) Modify “FSP_STARTUP_LOCATION” to “0x10080000” in “script/redefinition_memory_regions_gcc.h”

Core	Current	Modify
CR52_0/ CR52_1/	FSP_STARTUP_LOCATION = 0x10060000;	FSP_STARTUP_LOCATION = 0x10080000;

(2) Modify “ENABLE_BL2” to “1” in “/src/RM_OpenAMP_APP/platform_info.h”.

Core	Current	Modify
CA55_1/ CA55_2/ CA55_3	#define ENABLE_BL2 (0U)	#define ENABLE_BL2 (1U)

Note:

The ENABLE_BL2 flag is enabled to allow the use of SEMRAR0 (0x80240180). This register is used in BL2 to ensure that GIC-600 is properly configured before the secondary core (FSP core) is booted.

(3) In the Clocks tab configuration, navigate to **CLMA6 Enable** >> **CLMA6 Alternative CLK** and set it to **PLL**.

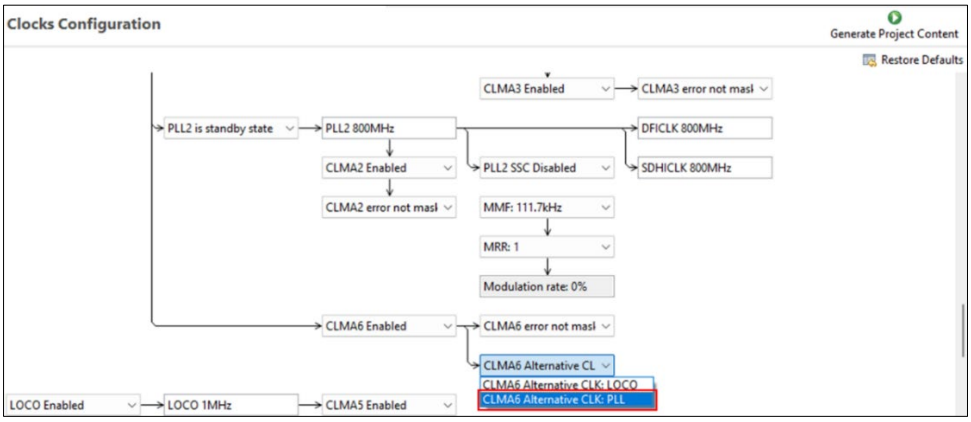


Figure 4-10. Clocks tab configuration

(4) Click the Generate Project Content button to complete the process

(5) Build the project by selecting **Choose Project** >> **Build Project**. The .srec file can be found in the Debug folder

(6) For program invocation, refer to Section **4.5.5 Multi-cores sample program invocation with BL2 of Trusted Firmware-A.**

4.4 Sample program set up on EWARM

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

(1) Extract **r01an8260ej0400-rz-multi-os-pkg.zip** on your development PC

(2) Extract the IAR project

The project for **FSP - FSP** combinations follows the naming format below:

<*device*>_<*platform_type*>_<*role*>_<*device*>_<*platform_type*>_<*role*>.zip

Where:

<device>: rzn2h_cr52_0 / rzn2h_cr52_1 / rzn2h_ca55_0 / rzn2h_ca55_1 / rzn2h_ca55_2 / rzn2h_ca55_3

<platform_type>: baremetal / freertos

<role>: master / slave

(e.g., rzn2h_ca55_0_baremetal_master_ca55_1_baremetal_slave.zip)

Meanwhile, the project for **Linux - FSP** combinations follows this format:

iar_<*device*>_rpmsg_linux_<*platform_type*>_demo.zip

Where:

<device>: rzn2h_cr52_0 / rzn2h_cr52_1 / rzn2h_ca55_1 / rzn2h_ca55_2 / rzn2h_ca55_3

<platform_type>: baremetal / freertos

<role>: master / slave

(e.g., gcc_rzn2h_ca55_1_rpmsg_linux_baremetal_demo.zip)

(3) Open EWARM and click **File** >> **Open Worksapce**

(4) Navigate to the folder containing the project extracted in step 1, select the IAR project, then click Open.

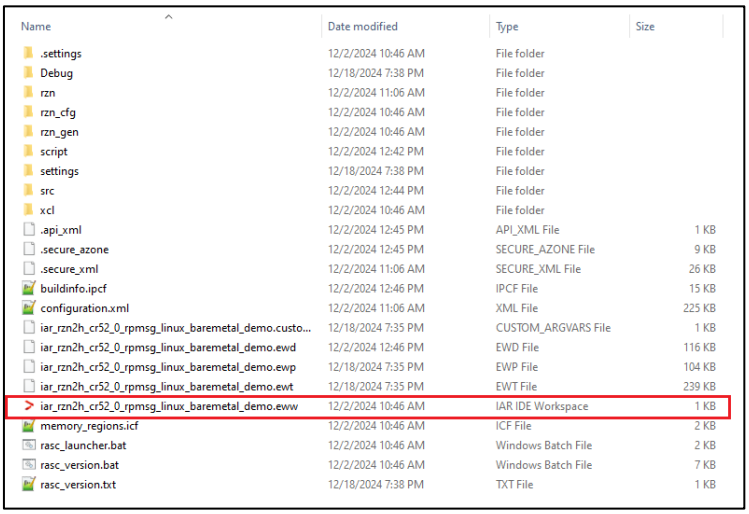


Figure 4-11. Import sample project on EWARM

Note:

1) For FSP - FSP combinations, since both cores use FSP, you need to open both projects, including one master and one slave, for the example.

2) For Linux - FSP combinations, since one core uses FSP and one core uses Linux, you only need to open one project for the example.

4.4.1 Setting for sample program to invocation with I-Jet

When invoking with I-Jet, perform the following step:

(1) Build the project from **Choose Project** >> **Rebuild All**

Note:

Refer to the Note in Section **4.3.1 Setting for Sample Program to Invocation with Segger J-Link** to see the project build requirements.

(2) For program invocation, refer to Section **4.5.2 Multi-cores sample program invocation with I-Jet on EWARM.**

(Optional setting for the remoteproc sample)

4.4.2 Setting for sample program to invocation with remotepoc

When invoking remoteproc on a Linux - FSP combination, perform the following step:

(1) Build the project from **Choose Project** >> **Rebuild All**

(2) For program invocation, refer to Section **4.5.3 Multi-cores sample program invocation with remoteproc.**

Note:

By default, the firmware for the remoteproc sample is pre-prepared in “/meta-rz-multi-os/recipes-firmware /cr52-firmware/files”. Therefore, rebuilding and reloading the sample is only necessary when modifications are made.

(Optional setting for the u-boot sample)

4.4.3 Setting for sample program to invocation with u-boot

When invoking u-boot on a Linux - FSP combination, perform the following step:

(1) In the Clocks tab configuration, navigate to **CLMA6 Enable** >> **CLMA6 Alternative CLK** and set it to **PLL**.

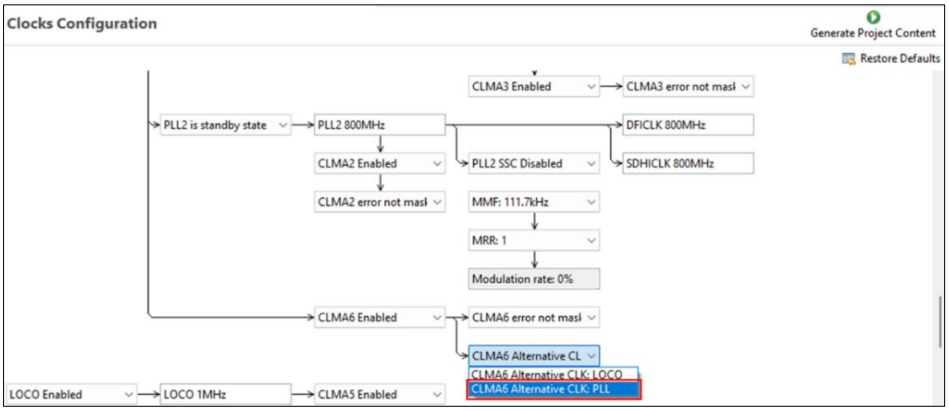


Figure 4-12. Clocks tab configuration

Note:

Please note that this configuration is performed on the primary core (CR52_0).

(2) Click the Generate Project Content button to complete the process.

(3) Navigate to Choose Project >> Options >> Output Converter >> Output, then set Output format to Raw binary. Click OK to complete.

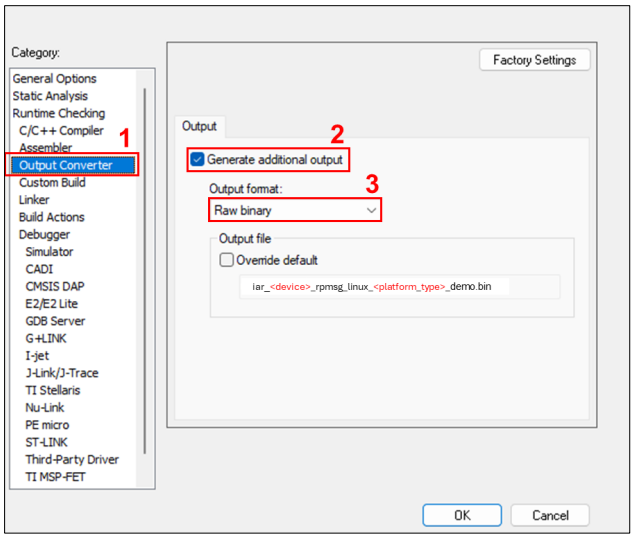


Figure 4-13. Firmware Binary Configuration on EWARM

(4) Build the project by selecting **Project** >> **Rebuild All**. The .bin file can be found in the Debug folder.

(5) For program invocation, refer to Section **4.5.4 Multi-cores sample program invocation with u-boot.**

(Optional setting for the BL2 of Trusted Firmware-A sample)

4.4.4 Setting for sample program to invocation with BL2 of Trusted Firmware-A

When invoking BL2 of Trusted Firmware-A on a Linux - FSP combination, perform the following step:

(1) Modify “FSP_STARTUP_LOCATION” to “0x10080000” in “script/redefinition_memory_regions_iar.h”.

Core	Current	Modify
CR52_0/ CR52_1/	FSP_STARTUP_LOCATION = 0x10060000;	FSP_STARTUP_LOCATION = 0x10080000;

(2) Modify “ENABLE_BL2” to “1” in “/src/RM_OpenAMP_APP/platform_info.h”.

Core	Current	Modify
CA55_1/ CA55_2/ CA55_3	#define ENABLE_BL2 (0U)	#define ENABLE_BL2 (1U)

Note:

The ENABLE_BL2 flag is enabled to allow the use of SEMRAR0 (0x80240180). This register is used in BL2 to ensure that GIC-600 is properly configured before the secondary core (FSP core) is booted.

(3) In the Clocks tab configuration, navigate to CLMA6 Enable >> CLMA6 Alternative CLK and set it to PLL.

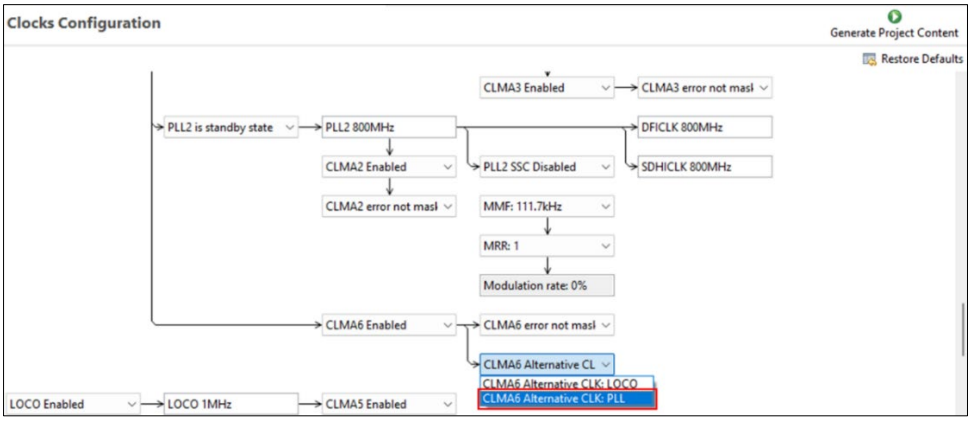


Figure 4-14. Clocks tab configuration

Note:
Please note that this configuration is performed on the primary core (CR52_0).

(4) Click the Generate Project Content button to complete the process.

(5) Build the project from **Choose Project** >> **Rebuild All**

(6) The .srec file can be found in the Debug folder. Converting them to the S3 record format is required for the firmware to work with BL2. Use tools such as objcopy, arm-none-eabi-objcopy, or aarch64-none-elf-objcopy to perform the conversion to the S3 record format.

```
objcopy.exe -I srec -O srec --srec-forceS3 input_s1.srec output_s3.srec
```

(7) For program invocation, refer to Section **4.5.5 Multi-cores sample program invocation with BL2 of Trusted Firmware-A.**

4.5 Multi-cores sample program invocation

4.5.1 Multi-Cores sample program invocation with Segger J-Link on e² studio

(1) Turn on the board

Configure the switches according to **Section 4.2.1 Hardware Setup When Using e2 studio** to select either the FSP - FSP or Linux - FSP combination.

If the Linux - FSP combination is selected, wait until the Linux boot process is complete and the message shown below appears before proceeding:

```
Poky (Yocto Project Reference Distro) 5.0.11 rzn2h-dev ttySC0
rzn2h-dev login:
```

(2) Click Debug button in Debug Configuration

If “A Renesas GDB debug session is already active” window below appears, please press “No” to go ahead.

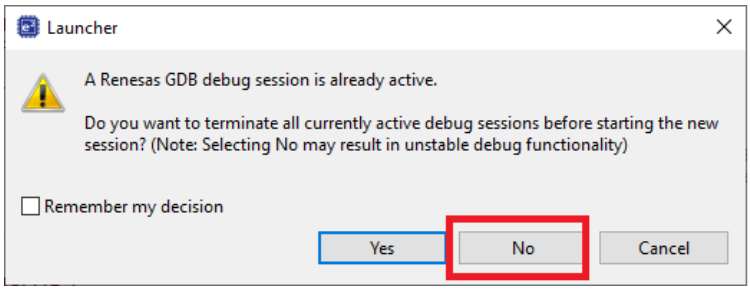


Figure 4-15. Renesas GDB debug session is already active

If “Proceed with lauch” window below appears, please press “yes” to go ahead.

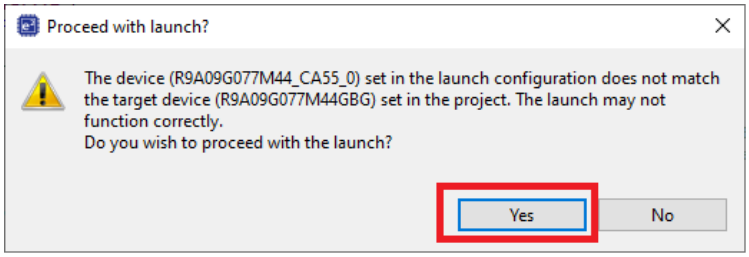


Figure 4-16. Proceed with launch

While in Debug mode, click **Run** >> **Resume** or click on the **Play** icon twice.

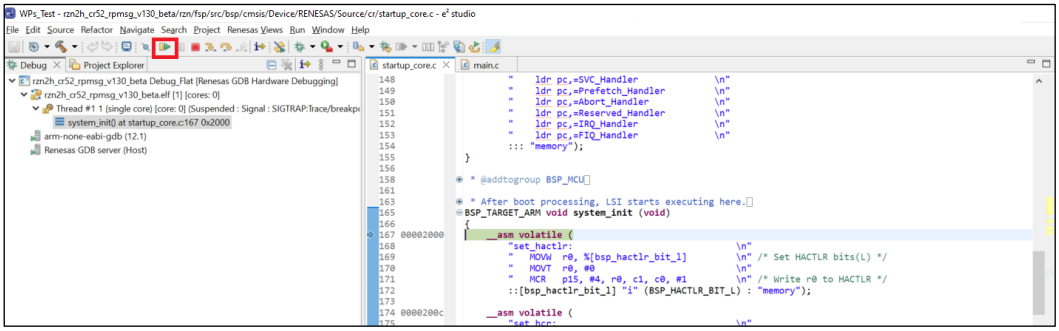


Figure 4-17. e² Studio Debugger Memory Window

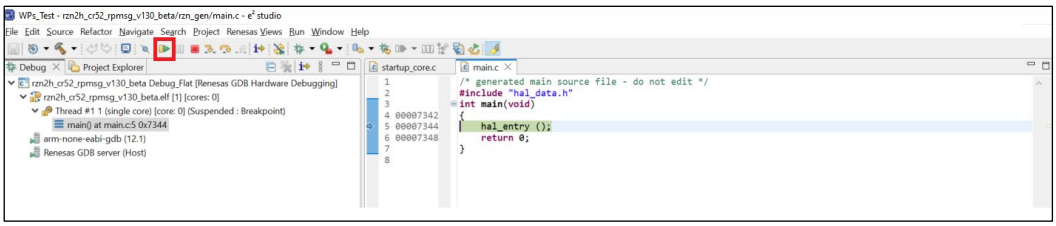


Figure 4-18. Hal entry

Note:

1) For FSP - FSP combinations:

- CR52_0 or CA55_0 must execute before other cores.
- In the case where CR52_0 or CA55_0 is neither a slave core nor a master core, rzn2h_cr52_0_boot /rzn2h_ca55_0_boot must run before other cores begin execution.
- In the case of CR52_0 or CA55_0 is master core, it should be setup running as procedure below:
  - Run the master core until it reaches the main function.
  - Running slave core programming.
  - Running master core programming again.
- Slave must run before running master as OpenAMP procedure.

2) For Linux- FSP combinations:

- For the sample program of the CR52_1/ CA55_1/ CA55_2/ CA55_3 cores, the rzn2h_cr52_0_boot project only needs to complete the project build process and does not need to be downloaded to the board.

(3) Run the multi-cores project

FSP – FSP combinations:

Console of Slave Core: OpenAMP opens successfully.

```
Successfully open uio device: E0000000.rsctbl.
Successfully added memory device E0000000.rsctbl.
Successfully open uio device: E1000000.vring-ctl0.
Successfully added memory device E1000000.vring-ctl0.
Successfully open uio device: E1200000.vring-shm0.
Successfully added memory device E1200000.vring-shm0.
Initialize remoteproc successfully.
creating remoteproc virtio
initializing rpmsg vdev
Remote proc init.
RPMSG endpoint has created.
```

Console of Master Core: OpenAMP opens successfully.

```
Successfully open uio device: E0000000.rsctbl.
Successfully added memory device E0000000.rsctbl.
Successfully open uio device: E1000000.vring-ctl0.
Successfully added memory device E1000000.vring-ctl0.
Successfully open uio device: E1200000.vring-shm0.
Successfully added memory device E1200000.vring-shm0.
Initialize remoteproc successfully.
creating remoteproc virtio
initializing rpmsg vdev
Remote proc init.
RPMSG endpoint has created.
```

Linux – FSP combinations:

Console of FSP Core: FSP program is waiting for the establishment of rpmsg channel between FSP and Linux.

```
***************  CR52_0 BAREMETAL  ****************
**************************************************
Successfully probed IPI device
Successfully open uio device: 3e0000000.rsctbl.
Successfully added memory device 3e0000000.rsctbl.
Successfully open uio device: 3e1000000.vring-ctl0.
Successfully added memory device 3e1000000.vring-ctl0.
Successfully open uio device: 3e1200000.vring-shm0.
Successfully added memory device 3e1200000.vring-shm0.
Initialize remoteproc successfully.
creating remoteproc virtio
initializing rpmsg vdev
```

(4) Sample application result

FSP – FSP combinations:

LED ON: Turn on by entering 43210C01 on the Master Core.

```
Successfully open uio device: E0000000.rsctbl.
Successfully added memory device E0000000.rsctbl.
Successfully open uio device: E1000000.vring-ctl0.
Successfully added memory device E1000000.vring-ctl0.
Successfully open uio device: E1200000.vring-shm0.
Successfully added memory device E1200000.vring-shm0.
Initialize remoteproc successfully.
creating remoteproc virtio
initializing rpmsg vdev
Remote proc init.
RPMSG endpoint has created.
43210C01
```

LED OFF: Turn off by entering 43210C00 on the Master Core.

```
Successfully open uio device: E0000000.rsctbl.
Successfully added memory device E0000000.rsctbl.
Successfully open uio device: E1000000.vring-ctl0.
Successfully added memory device E1000000.vring-ctl0.
Successfully open uio device: E1200000.vring-shm0.
Successfully added memory device E1200000.vring-shm0.
Initialize remoteproc successfully.
creating remoteproc virtio
initializing rpmsg vdev
Remote proc init.
RPMSG endpoint has created.
43210C00
```

Note:

If you want the input data (e.g., “43210C01” or “43210C00”) to be displayed in Tera Term, enable the Local Echo option under [Setup] >> [Terminal] in Tera Term, as shown in Figure 3-12.

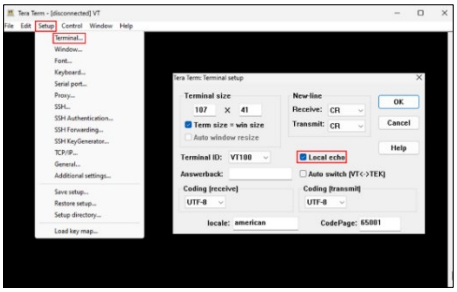


Figure 4-19. Enabling Local Echo to display input data in Tera Term

Linux – FSP combinations: Run Linux sample program by executing the following command on Linux.

For the sample program for FSP (CR52) – Linux, execute the command as below:

```
rzn2h-dev login: root
root@rzn2h-dev:~# rpmsg_sample_client 0
```

For the sample program for FSP (CA55) – Linux, follow the steps below:

- After powering on the board, during the countdown process, press any key to stop.

```
NOTICE:  BL2: v2.7(release):2.7.0/t2h_n2h_1.0.2-dirty
NOTICE:  BL2: Built : 17:00:45, Feb 28 2025
NOTICE:  BL2: Booting BL31
NOTICE:  BL31: v2.7(release):2.7.0/t2h_n2h_1.0.2-dirty
NOTICE:  BL31: Built : 17:00:45, Feb 28 2025
U-Boot 2021.10 (Mar 13 2025 - 09:27:21 +0000)
CPU: Renesas Electronics RZ/N2H
Model: Renesas Development EVK based on r9a09g087m44
DRAM:  7.9 GiB
MMC:   mmc@92080000: 0, mmc@92090000: 1
Loading Environment from MMC... OK
In:    serial@80005000
Out:   serial@80005000
Err:   serial@80005000
Net:
Warning: ethernet@92010000 (eth1) using random MAC address - 52:e8:97:ee:ef:2f
Warning: ethernet@92000000 (eth0) using random MAC address - be:52:2e:57:bd:85
eth0: ethernet@92000000, eth1: ethernet@92010000
Hit any key to stop autoboot:  2
=>
```

- Enter the commands below to support the corresponding CA55 cores.

```
=> setenv bootargs 'ignore_loglevel console=ttySC0,115200n8 rootwait
root=/dev/mmcblk1p2 earlycon'
=> setenv bootcmd 'ext4load mmc 1:2 0xC4200000 boot/Image; ext4load mmc 1:2
0xC5F00000 boot/r9a09g087m44-dev-fsp-<a55_core>-support.dtb; booti 0xC4200000
- 0xC5F00000'
=> boot
```

<a55_core>: a551 (for CA55_1 core)/ a552 (for CA55_2 core)/ a553 (for CA55_3 core)

- Execute the command below.

```
root@rzn2h-dev:~# rpmsg_sample_client 0
```

Then, you can see the following message on the console of FSP and Linux

Below is an example console output for the combination FSP (CR52) – Linux.

Console of FSP (CR52):

```
***************  CR52_0 BAREMETAL  ****************
**************************************************
Successfully probed IPI device
Successfully open uio device: 3e0000000.rsctbl.
Successfully added memory device 3e0000000.rsctbl.
Successfully open uio device: 3e1000000.vring-ctl0.
Successfully added memory device 3e1000000.vring-ctl0.
Successfully open uio device: 3e1200000.vring-shm0.
Successfully added memory device 3e1200000.vring-shm0.
Initialize remoteproc successfully.
creating remoteproc virtio
initializing rpmsg vdev
De-initializating remoteproc
creating remoteproc virtio
initializing rpmsg vdev
```

Console of Linux:

```
echo test: sent : 484
 received payload number 467 of size 484
sending payload number 468 of size 485
echo test: sent : 485
 received payload number 468 of size 485
sending payload number 469 of size 486
echo test: sent : 486
 received payload number 469 of size 486
sending payload number 470 of size 487
echo test: sent : 487
 received payload number 470 of size 487
sending payload number 471 of size 488
echo test: sent : 488
 received payload number 471 of size 488
**********************************
 Test Results: Error count = 0
**********************************
Quitting application .. Echo test end
Stopping application...
root@rzn2h-dev:~#
```

4.5.2 Multi-cores sample program invocation with I-Jet on EWARM

(1) Turn on the board

Configure the switches according to Section **4.2.2 Hardware Setup When Using EWARM** to select either the FSP - FSP or Linux - FSP combination.

If the Linux - FSP combination is selected, wait until the Linux boot process is complete and the message shown below appears before proceeding:

```
Poky (Yocto Project Reference Distro) 5.0.11 rzn2h-dev ttySC0
rzn2h-dev login:
```

(2) Click the Download and Debug button on the toolbar to enter debug mode

Figure 4-20. Config for debug

Note:

Refer to the Note in step **(2) Click Debug button in Debug Configuration** in Section **4.5.1 Multi-Cores Sample Program Invocation with Segger J-Link on e² studio**.

(3) Run the multi-cores project

Refer to step **(3) Run the multi-cores project** in Section **4.5.1 Multi-Cores Sample Program Invocation with Segger J-Link on e² studio**.

(4) Sample application result

Refer to step **(4) Sample application result** in Section **4.5.1 Multi-Cores Sample Program Invocation with Segger J-Link on e² studio**

4.5.3 Multi-cores sample program invocation with remoteproc

On **RZ/N2H** evaluation board, you can invoke RPMsg sample program with remoteproc by following the procedure stated below:

(1) Booting up Linux by following Linux Start-up Guide

(2) Invoke the command below to specify RPMsg sample program to be loaded

```
root@rzn2h-dev:~# echo
<env>_<device>_rpmsg_linux_<platform_type>_demo.<extension>
> /sys/class/remoteproc/<remoteprocX>/firmware
```

<env>: gcc/ iar.

<device>: rzn2h_cr52_0/ rzn2h_cr52_1.

<platform_type>: baremetal/ freertos.

<extension>: elf (for GCC)/ out (for IAR).

<remoteprocX>: remoteproc0 (for kick CR52_0)/ remoteproc1 (for kick CR52_1).

Note:

If the Linux BSP was previously built using remoteproc from an older version of the Multi-OS Package, the latest remoteproc sample program may not be updated. The firmware files are located in /lib/firmware on the Micro-SD card, so you can check them there. To ensure the newest updates are applied, please delete the “build” directory located in the working directory and rebuild the Linux BSP from scratch.

(3) Kick CR52 by using the command below

```
root@rzn2h-dev:~# echo start > /sys/class/remoteproc/remoteprocX/state
```

If CR52 starts to work successfully, the following message should be shown:

```
root@rzn2h-dev:~# echo start > /sys/class/remoteproc/remoteprocX/state
[44.818601] remoteproc remoteprocX: powering up <core>
[44.905556] remoteproc remoteprocX: Booting fw image <image>, size <size>
[44.915926] remoteproc remoteprocX: unsupported resource 4
[44.947845]  remoteprocX#vdev0buffer:
assigned reserved memory node vdev0buffer@0x3E1200000
[44.956469]  remoteprocX#vdev0buffer: registered virtio0 (type 7)
[44.962655] remoteproc remoteprocX: remote processor <core> is now up
```

After kick CR52, RPMsg sample can run by using the command below:

```
root@rzn2h-dev:~# rpmsg_sample_client 0
```

(4) Stop CR52 by using the command below

```
root@rzn2h-dev:~# echo stop > /sys/class/remoteproc/remoteprocX/state
```

If CR52 stops to work successfully, the following message should be shown:

```
root@rzn2h-dev:~# echo stop > /sys/class/remoteproc/remoteprocX/state
[909.395289] remoteproc remoteprocX: stopped remote processor <core>
```

(5) Restart CR52

After stopping CR52, if you want to restart it, execute the command below:

```
root@rzn2h-dev:~# echo start > /sys/class/remoteproc/remoteprocX/state
```

(6) Change the firmware and shut down the board

Press the reset button to reboot Linux and repeat steps (1) to (5) if you need to change the firmware.

Run the shutdown command on the console as shown below to safely power off the board after the program has finished running. After executing the shutdown command, you will see the "reboot: Power down" message. Then, turn off the POWER_SW.

```
root@rzn2h-dev:~# shutdown -h now
```

(Optional for firmware allocated in the TCM region)

(7) Change the FSP project to allocate memory in the TCM region

The default remoteproc sample program operates in the System RAM region. You can make the following changes in the FSP project to switch it to the TCM region.

- GCC project:

Remove “FSP_STARTUP_LOCATION = 0x10060000;” in the “script/redefinition_memory_regions_gcc.h” file.

Core	Current	Modify
CR52_0/ CR52_1	/* User customizes the location in here */ FSP_STARTUP_LOCATION = 0x10060000;	/* User customizes the location in here */ // FSP_STARTUP_LOCATION = 0x10060000;

- IAR project:

Remove “FSP_STARTUP_LOCATION = 0x10060000;” in the “script/redefinition_memory_regions_iar.h” file.

Core	Current	Modify
CR52_0/ CR52_1	/* User customizes the location in here * define symbol FSP_STARTUP_LOCATION_ADDRESS = 0x10060000;	/* User customizes the location in here * define symbol // FSP_STARTUP_LOCATION_ADDRESS = 0x10060000;

After modifying the project, build the CR52 project to obtain the .elf (GCC) or .out (IAR) file from the Debug folder.

(8) Modify the Device Tree in the Linux BSP to boot CR52 in the TCM region

When booting CR52 in the TCM region, please modify the property “renesas,rz-start_address = <0x00000000>;” in the cr52_0_rproc/ cr52_1_rproc node within the patch file “0003-dts-renesas-Add-rproc-node-for-CR52.patch”, located at meta-rz-multi-os/recipes-kernel/linux/linux-renesas.

Core	Current	Modify
CR52_0	+ cr52_0_rproc: cr52_0 { ..... + renesas,rz-core = <0x0>; + renesas,rz-swint = <10>; + renesas,rz-rsctbl = <0x30xE0000000>; + renesas,rz-start_address = <0x10060000>; ..... + };	+ cr52_0_rproc: cr52_0 { ..... + renesas,rz-core = <0x0>; + renesas,rz-swint = <10>; + renesas,rz-rsctbl = <0x30xE0000000>; + renesas,rz-start_address = <0x00000000>; ..... + };
CR52_1	+ cr52_1_rproc: cr52_1 { ..... + renesas,rz-core = <0x1>; + renesas,rz-swint = <11>; + renesas,rz-rsctbl = <0x30xE0000000>; + renesas,rz-start_address = <0x10060000>; ..... +};	+ cr52_1_rproc: cr52_1 { ..... + renesas,rz-core = <0x1>; + renesas,rz-swint = <11>; + renesas,rz-rsctbl = <0x30xE0000000>; + renesas,rz-start_address = <0x00000000>; ..... +};

After modifying the patch file, rebuild the Linux BSP to apply the changes using the command below. Once the build is complete, re-prepare the SD card with the updated core-image-minimal-rzn2h-dev.wic.gz.

```
$ MACHINE=rzn2h-dev bitbake core-image-minimal
```

(9) Integrate the firmware into Linux rootfs

After obtaining the CR52 firmware, you can integrate it into the Linux rootfs by copying the firmware to the “/lib/firmware” directory using the following steps.

(1) Insert the Micro-SD card into your Linux PC.

(2) Check the device name and mount with partition 2 of Micro-SD card.

Run the command below to determine the device name assigned to the Micro-SD card. In this example, it's “/dev/sdb”. If necessary, replace “/dev/sdb” with the correct device name for your system. After that, mount with partition 2 of Micro-SD card.

```
$ sudo fdisk -l
Disk /dev/sdb: 59,49 GiB, 63864569856 bytes, 124735488 sectors
Disk model: Transcend
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
```

```
Disk identifier: 0x8c4022be
Device Boot Start End Sectors Size Id Type
/dev/sdb1 2048 1050623 1048576 512M c W95 FAT32 (LBA)
/dev/sdb2 1050624 4172703 3122080 1,5G 83 Linux $
$ sudo mount /dev/sdb2 /media/
```

(3) Copy CR52 firmware from user path in Linux PC to “/lib/firmware” in Micro-SD card.

```
$ cd <user path>
$ sudo cp <env>_<device>_rpmsg_linux_<platform_type>_demo.<extension> \
 /media/lib/firmware
```

(10) Invoke the new firmware from remoteproc

After copying, verify that the firmware exists in the “/lib/firmware” directory on the Micro-SD card, then unmount the Micro-SD card.

```
$ sudo umount /media/
```

To invoke the sample, follow steps (1) to (6) in this section.

4.5.4 Multi-cores sample program invocation with u-boot

On **RZ/N2H** evaluation board, you can invoke RPMsg sample program with u-boot by following the procedure stated below:

(1) Place the binary file into the first partition of the Micro-SD card

Run the command below to determine the device name assigned to the micro-SD card. In this example, it's "/dev/sdb". If necessary, replace "/dev/sdb" with the correct device name for your system. After that, mount with partition 1 of SD card.

```
$ sudo fdisk -l
Disk /dev/sdb: 59,49 GiB, 63864569856 bytes, 124735488 sectors
Disk model: Transcend
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x8c4022be
Device Boot Start End Sectors Size Id Type
/dev/sdb1 2048 1050623 1048576 512M c W95 FAT32 (LBA)
/dev/sdb2 1050624 4172703 3122080 1,5G 83 Linux $
$ sudo mount /dev/sdb1 /media/
```

Copy binary file from user path in Linux PC to first partition in SD card.

```
$ cd <user path>
$ sudo cp /<env>_<device>_rpmsg_linux_<platform_type>_demo.bin \
 /media/
```

<env>: gcc/ iar.

<device>: rzn2h_cr52_0/ rzn2h_cr52_1.

<platform_type>: baremetal.

Note: Please adjust the paths of the binary file and bootloaderf2 to match the actual paths in your environment.

(2) Insert Micro-SD card to RZ/N2H evaluation board

(3) Turn on the RZ/N2H board using the POWER_SW. To start invoking the sample from U-Boot, press the Reset button. Then, you should see the following message on the console

```
NOTICE:  BL2: v2.7(release):2.7.0/t2h_n2h_1.0.2-dirty
NOTICE:  BL2: Built : 17:00:45, Feb 28 2025
NOTICE:  BL2: Booting BL31
NOTICE:  BL31: v2.7(release):2.7.0/t2h_n2h_1.0.2
NOTICE:  BL31: Built : 17:00:45, Feb 28 2025
U-Boot 2021.10 (Mar 13 2025 - 09:27:21 +0000)
CPU: Renesas Electronics RZ/N2H
Model: Renesas Development EVK based on r9a09g087m44
DRAM:  7.9 GiB
```

```
MMC:   mmc@92080000: 0, mmc@92090000: 1
Loading Environment from MMC... Card did not respond to voltage select! : -84
*** Warning - No block device, using default environment
In:    serial@80005000
Out:   serial@80005000
Err:   serial@80005000
Net:
Warning: ethernet@92010000 (eth1) using random MAC address - 1a:b6:fa:3f:12:31
Warning: ethernet@92000000 (eth0) using random MAC address - 56:af:cf:b9:61:24
eth0: ethernet@92000000, eth1: ethernet@92010000
Hit any key to stop autoboot:  2
=>
```

(4) Hit any key within 3 sec to stop autoboot

(5) Carry out the following setup on u-boot to kick CR52

```
=> fatload mmc 1:1 0xC6001000
<env>_<device>_rpmsg_linux_<platform_type>_demo.bin
=> cr52 start_<core> 0x10060000 0x80000
=> boot
```

<env>: gcc/ iar.

<device>: rzn2h_cr52_0/ rzn2h_cr52_1.

<platform_type>: baremetal.

<core>: cr520/ cr521.

After kick CR52, RPMsg sample can run by using the command below:

```
root@rzn2h-dev:~# rpmsg_sample_client 0
```

(6) Change the firmware and shut down the board

Press the reset button to reboot Linux and repeat steps (1) to (5) if you need to change the firmware.

Run the shutdown command on the console as shown below to safely power off the board after the program has finished running. After executing the shutdown command, you will see the "reboot: Power down" message. Then, turn off the POWER_SW.

```
root@rzn2h-dev:~# shutdown -h now
```

4.5.5 Multi-cores sample program invocation with BL2 of Trusted Firmware-A

On **RZ/N2H** evaluation board, you can invoke RPMsg sample program with BL2 of Trusted Firmware-A by following the procedure stated below:

(1) Program the FSP image using FlashWriter

After rewriting the bootloader files (bl2_bp_xspi0-rzn2h-dev.srec and fip-rzn2h-dev.srec) , proceed to program the FSP image here.

```
> XSPIW 0 0x00200000 0x00060000    # input command
send file
```

Send file > "<env>_<device>_rpmsg_linux_<platform_type>_demo.srec"

```
xSPI Initialize complete
Erased
Writen
```

<env>: gcc/ iar.

<device>: rzn2h_cr52_0/ rzn2h_cr52_1/ rzn2h_ca55_1/ rzn2h_ca55_2/ rzn2h_ca55_3.

<platform_type>: baremetal.

After completing the process, turn off the board using the POWER_SW.

(2) Refer Linux Start-up Guide to boot the board

The FSP image will be loaded by the bootloader at BL2 of Trusted Firmware-A.

Note:

Please follow the instructions in Section 4.5.1 Run the Linux sample program by executing the following command on Linux for the corresponding combination: FSP (CR52) - Linux or FSP (CA55) - Linux.

5. Instructions for creating a new project (for unavailable combinations)

* Notes and constraints when creating new FSP – FSP combinations that is not supported by the sample projects.

(1) In a combination, **there must be at least one primary core (CR52_0/CA55_0 core)** with a role that can be either a slave or a master core. If neither the slave nor the master core you want to combine includes a primary core, you will need to create a boot project.

Note: A boot project is a project created with the CR52_0 or CA55_0 core.

(2) When starting the creation process, make sure that the primary core project is created first and successfully built in order to generate the smartbundle file for other core projects to use.

Note: The second core project must be linked to the smartbundle file of the first core project. Similarly, the third core project needs to link to the smartbundle file of the second core project, and so on, in the following order: CR52_0, CR52_1, CA55_0, CA55_1, CA55_2, CA55_3.

Example: Creating a combination with CR52_1 (slave) and CA55_1 (master).

- There is no primary core in the combination, so a boot project needs to be created.
- If you choose the CR52_0 core as the boot project, the project order will be: CR52_0 → CR52_1 → CA55_1.
- If you choose the CA55_0 core as the boot project, the project order will be: CA55_0 → CR52_1 → CA55_1.

5.1 Guidelines on e² studio

5.1.1 Baremetal project on e² studio

Refer to the Getting Started document, section 6. FSP Configuration Users Guide, to create a project in e² studio.

(1) Notes when creating a new project

- FSP version: Select the version corresponding to the current Multi-OS Package.
- Board: Select “RZN2H Evaluation Board (RAM execution without flash memory)”.
- Device: Select “R9A09G087M44GBG”.
- Toolchains: Select "GNU ARM Embedded" version "<*GNU Toolchain version*>” if using the CR52 core, or select "GCC ARM A-Profile (AArch64 bare-metal)" version "<*GCC Toolchain version*>" if using the CA55 core.

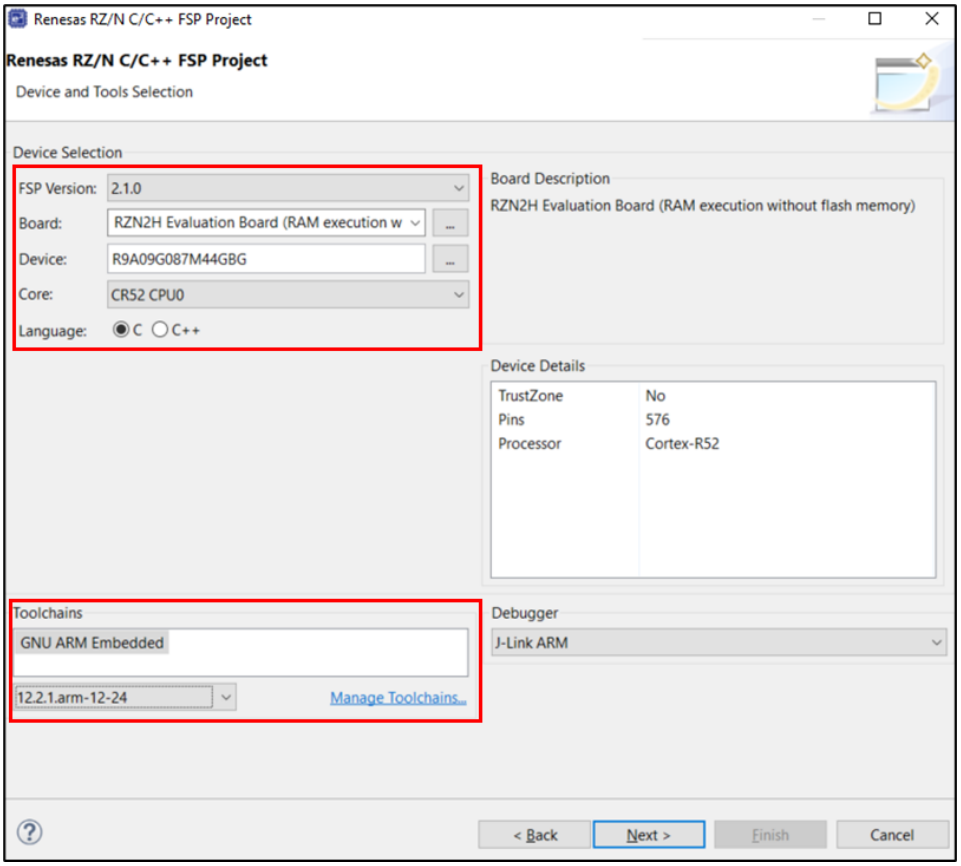


Figure 5-1. Device and Toolchain configuration

- Preceding Project or Smart Bundle Selection: Select "None" if the core project is the primary core or boot project. Select “Preceding Project” in all other cases.

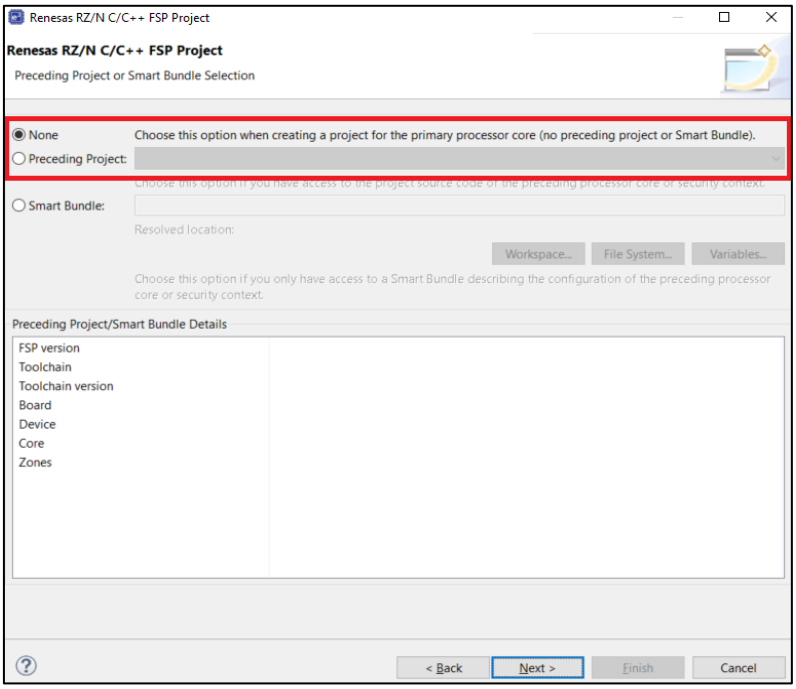


Figure 5-2. Preceding Project or Smart Bundle configuration

Note:

Except for the CA55_0 core, its role must be considered carefully. If CA55_0 is not the primary core (in the combination of CR52_0 core and CA55_0 core), select “Preceding Project” and link to the CR52_0 project in the dropdown list.

- Build Artifact and RTOS Selection: Select "No RTOS".

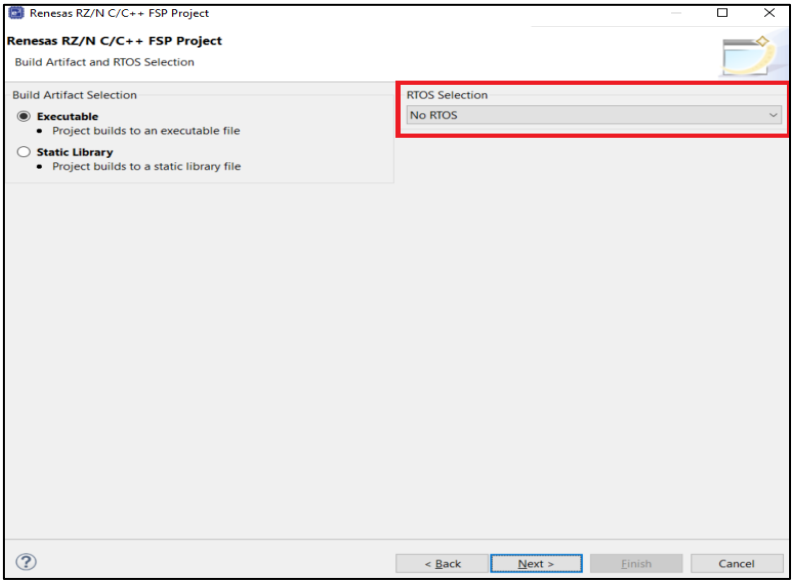


Figure 5-3. Build Artifact and RTOS configuration

- Project Template Selection: Select “Bare Metal – Minimal”.

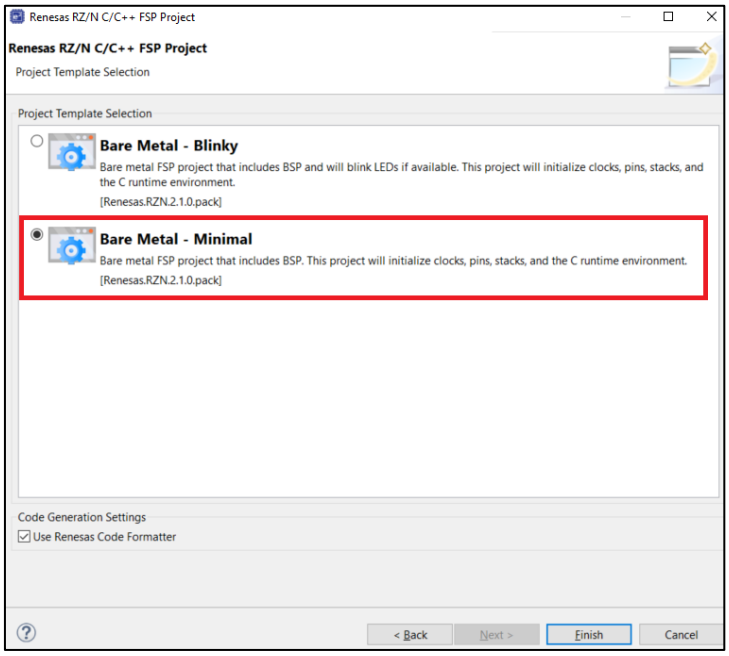


Figure 5-4. Project Template configuration

(2) Configuration in the configuration.xml file

- Stacks tab: Add OpenAMP, Inter-CPU IRQ, and the UART driver. Configure the drivers for the slave core and master core as shown below.

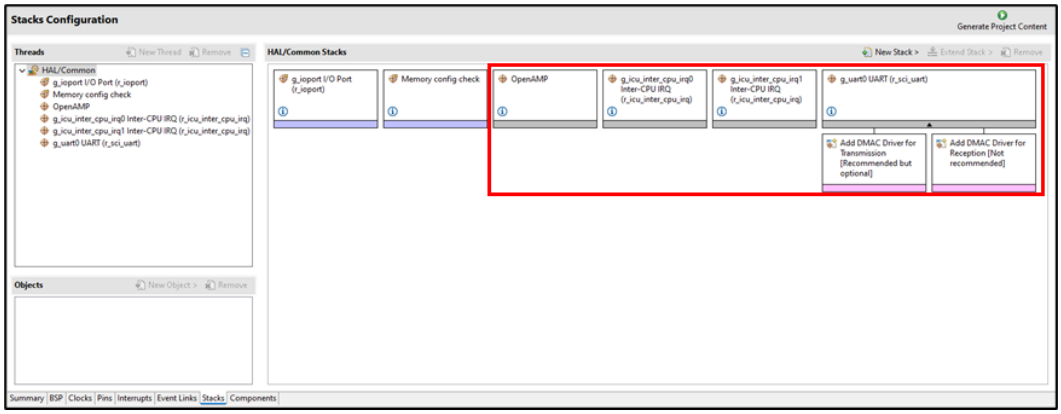


Figure 5-5. Stacks tab configuration

Inter-CPU IRQ			
Element	**Master core**	**Slave core**	**Remarks**
Parameter checking	Enable	Enable	-
g_icu_intercpu_irq0(channel)	1	0	-
g_icu_intercpu_irq1(channel)	0	1	-
g_icu_intercpu_irq0(Callback)	NULL	NULL	-
g_icu_intercpu_irq1(Callback)	rzn2_callback	rzn2_callback	-
g_icu_intercpu_irq0 (Software Interrupt Priority)	Disable	Disable	-
g_icu_intercpu_irq1 (Software Interrupt Priority)	30	30	-

UART			
Element	**Master core**	**Slave core**	**Remarks**
Parameter checking	Enable	Enable	-
Channel	1	0	-
Callback	uart_callback	uart_callback	-
Receive Interrupt Priority	30	30	-
Transmit Data Empty Interrupt Priority	30	30	-
Transmit End Interrupt Priority	30	30	-
Error Interrupt Priority	30	30	-

- BSP tab: Select "Enable" for DDR Initialization and Zero-initialized for DDR memory.

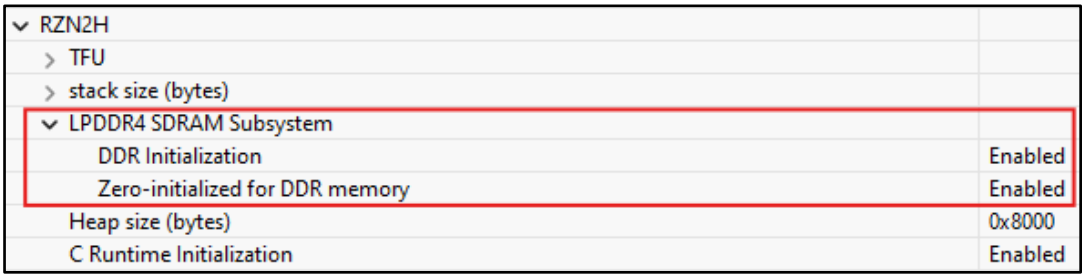


Figure 5-6. BSP tab configuration (1)

Note:

This configuration is only for the primary core.

- BSP tab: Select "Outer Shareable" for the Shareability field and choose "Attribute 3" for the Attribute Index for DDR mirror1 (for CR52 core) or DDR mirror (for CA55 core).

Figure 5-7. BSP tab configuration (2)

- Clocks tab: Change from "PLL2 is standby state" to "PLL2 is released from standby state".

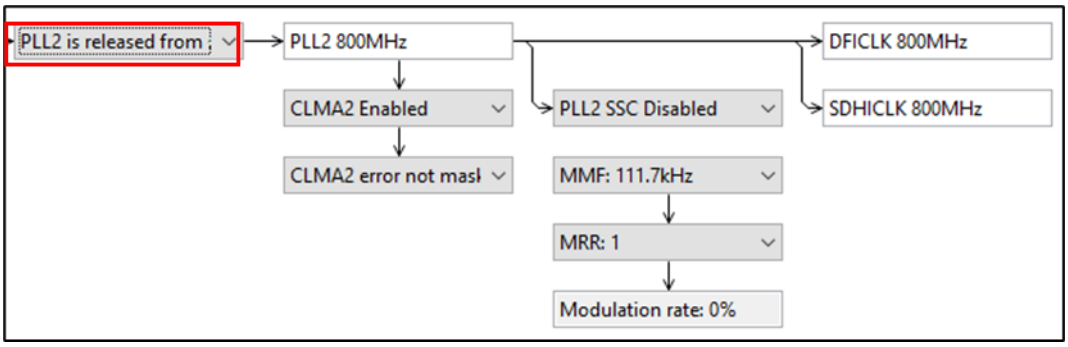


Figure 5-8. Clocks tab configuration

Note:

This configuration is only for the primary core.

- After completing the corresponding configurations above, click the Generate Project Content button.

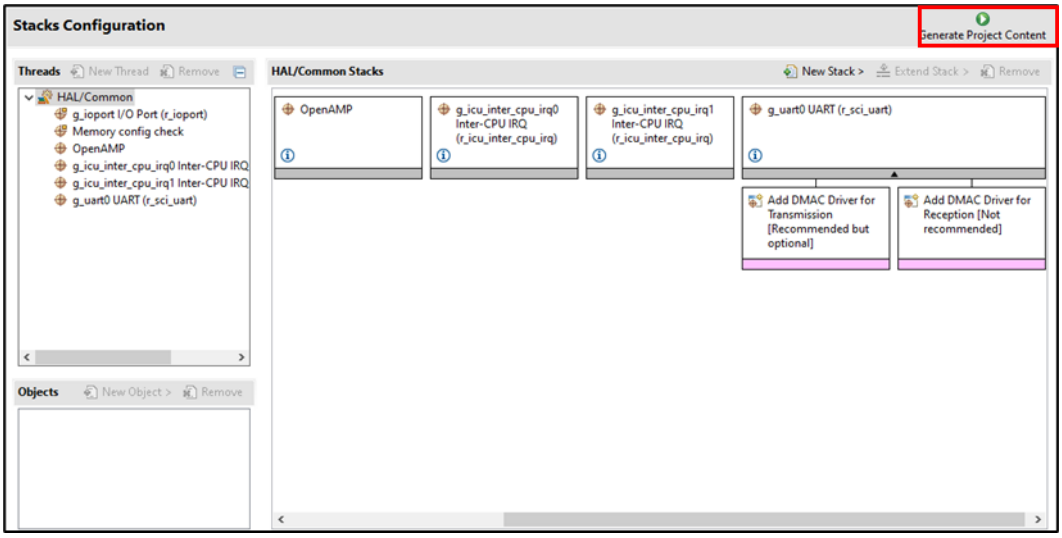


Figure 5-9. Generate project

(3) Copy the Baremetal folder

- The master core project, copy the files in the “Baremetal” folder from "Lib_mastercore" into the “src“ directory of the project.

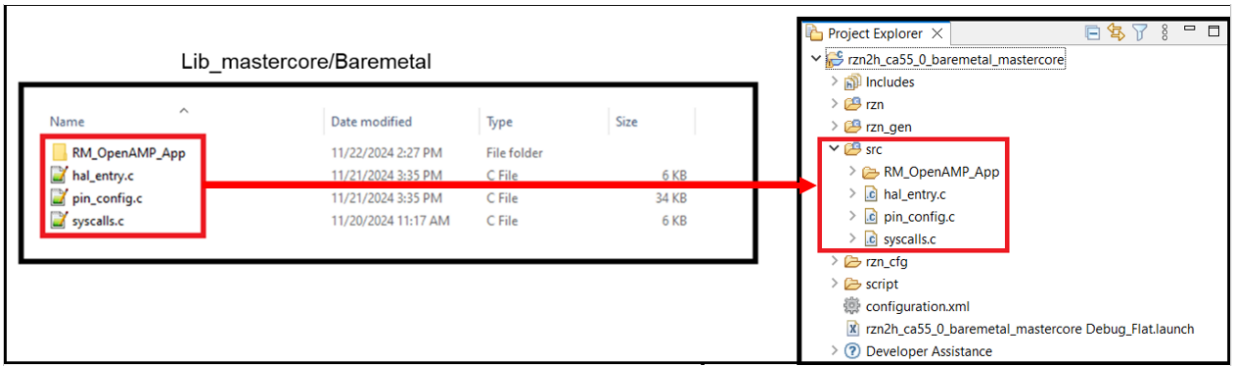


Figure 5-10. Copy the Baremetal folder from Lib_mastercore into the “src“ directory

- The slave core project, copy the files in the “Baremetal” folder from "Lib_slavecore" into the “src” directory of the project.

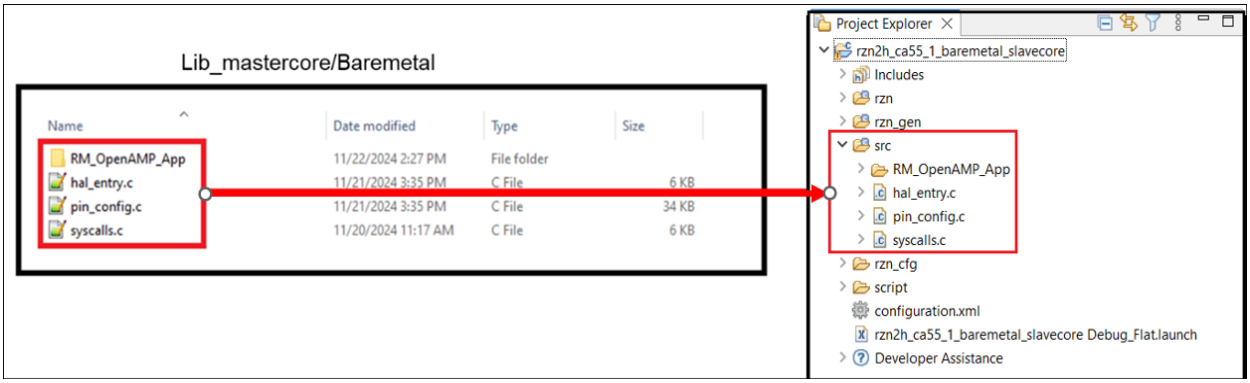


Figure 5-11. Copy the Baremetal folder from Lib_slavecore into the “src“ directory

- The boot project, copy the files in the “Lib_bootproject” folder into the “src” directory of the project.

- Copy and overwrite the files in the Patch folder into the corresponding paths for both master project and slave project as shown below.

Name file	Path to folder
redefinition_memory_regions_gcc.h	script/
fsp_ram_execution.ld	script/

(4) Build combination

Refer to **4.3.1 Setting for sample program to invocation with Segger J-Link**.

(5) Configuration the Debug for combination

- Clicking Run >> Debugger Configurations or by selecting the dropdown menu next to the bug icon and selecting Debugger Configurations.

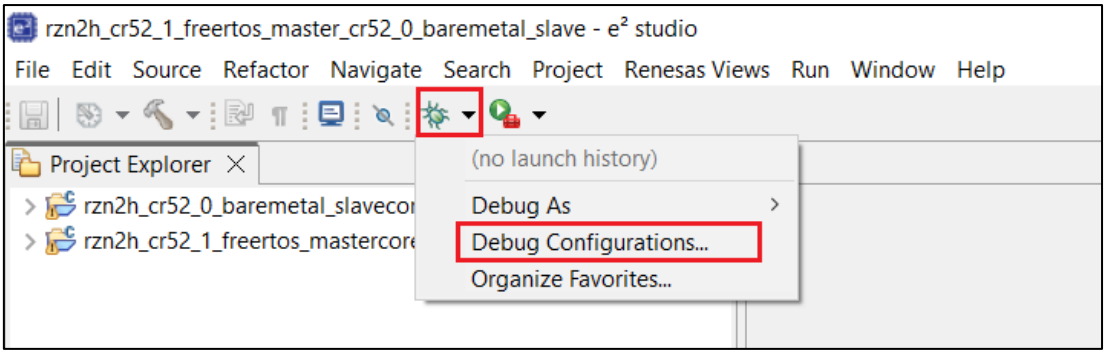


Figure 5-12. Select of Debug Configuration

- In the window Debug Configurations double click on "*Renesas GDB Hardware Debugging*" to create new configurations.

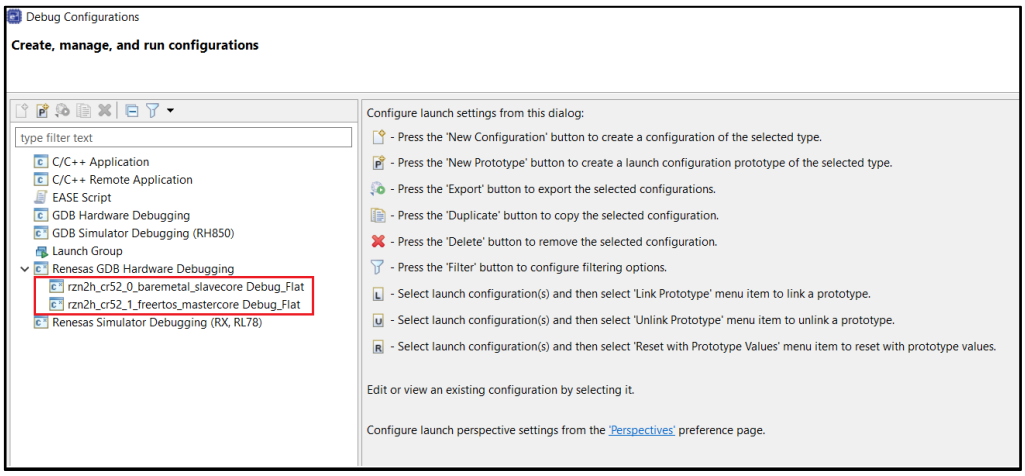


Figure 5-13. Configuration for debugging

- Main tab: Set rzn2h_<core>_<os>_mastercore and rzn2h_<core>_<os>_slavecore to Project with Browser button.

  Where:

  <core>: ca55_3/ ca55_2/ ca55_1/ ca55_0/ cr52_1/ cr52_0

  <os>: baremetal/ freertos

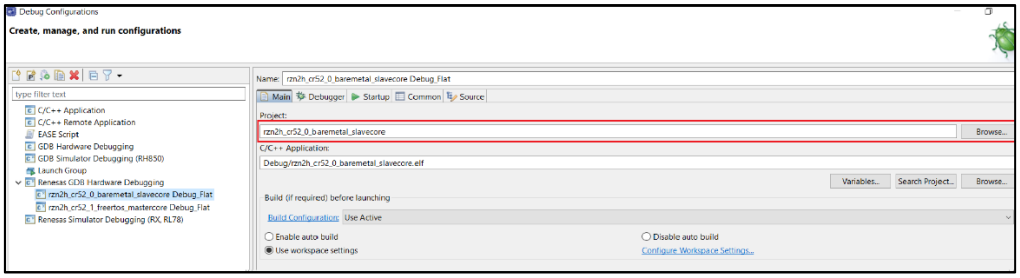


Figure 5-14. Configuration for debugging

- **Debugger tab >> GDB Settings:**
  - Debug hardware select **J-Link ARM**.
  - Target Device select **RZ** >> **RZ/RZN2H** >> **R9A07G087M44**_<core>.

  Where:

  <core>: CR52_0/ CR52_1/ CA55_0/ CA55_1/ CA55_2/ CA55_3

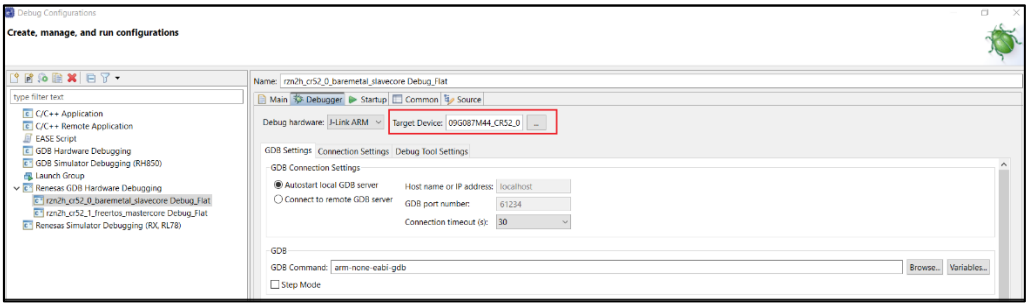


Figure 5-15. Configuration for target device

- **Debugger tab >> Connection Settings:**
  - **Set Reset at the beginning of connection** to **No**.
  - **Set CPSR(5bit) after download** to **Yes** (only for CR52_0 core).
  - Add "**${workspace_loc:/${ProjName}}/script/initialization_TCM.JlinkScript**" to the **Script File** field (only for the CR52_1 core).

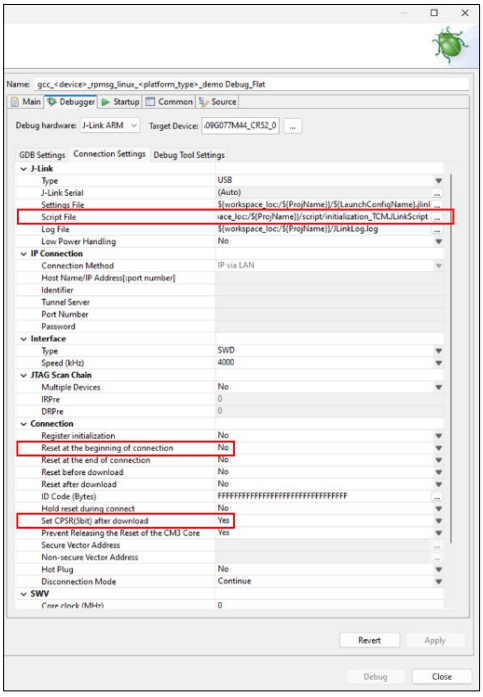


Figure 5-16. Debug Configuration

- Click "*Apply*" then "*Debug*" button in Debug Configuration.

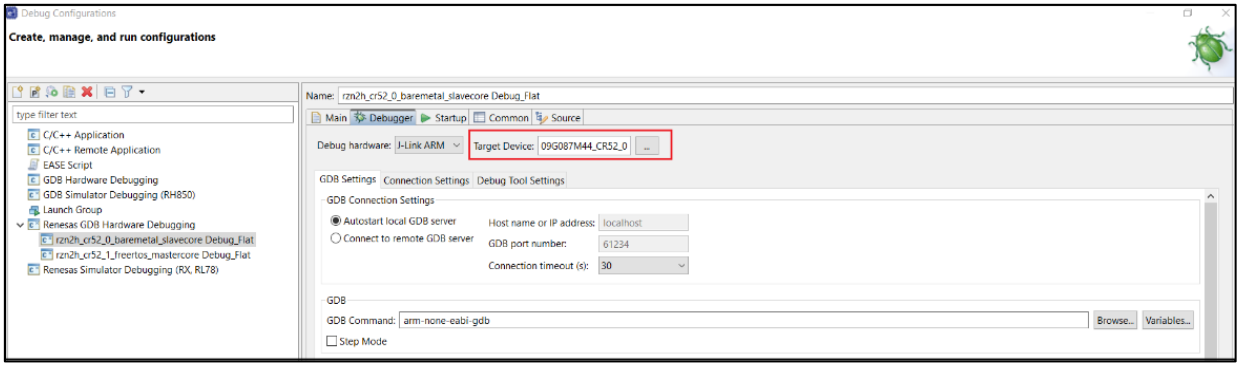


Figure 5-17. Save and run the debugging configuration

After completing the debug configuration, refer to Section 4.5 "Multi-Cores Sample Program Invocation" to run the example.

5.1.2 FreeRTOS project on e² studio

Refer to Section 5.1.1 “Baremetal Project on e² studio” to create the project with the following notes:

- Build Artifact and RTOS Selection: Select "FreeRTOS".

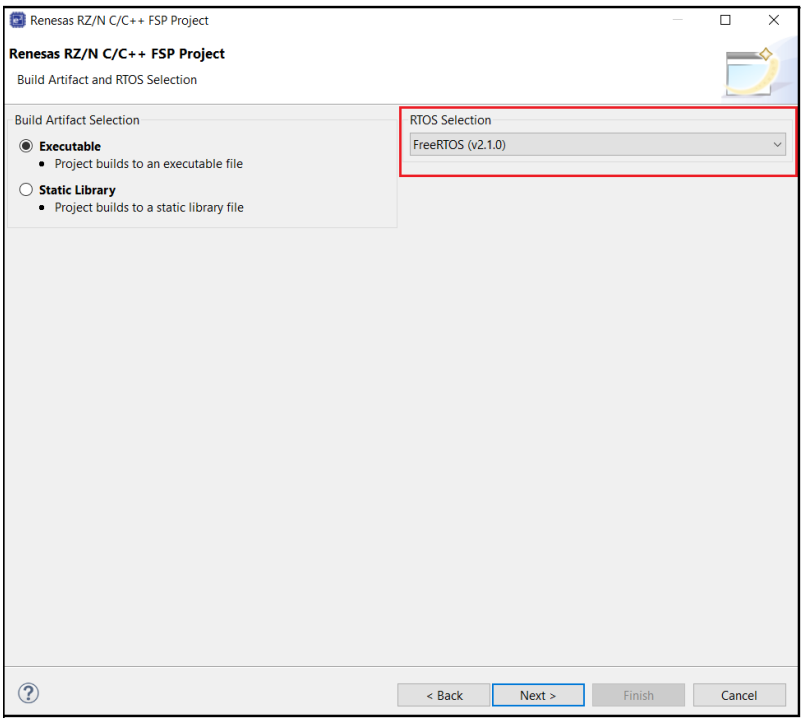


Figure 5-18. Build Artifact and RTOS configuration

- Project Template Selection: Select “FreeRTOS – Minimal – Static Allocation”.

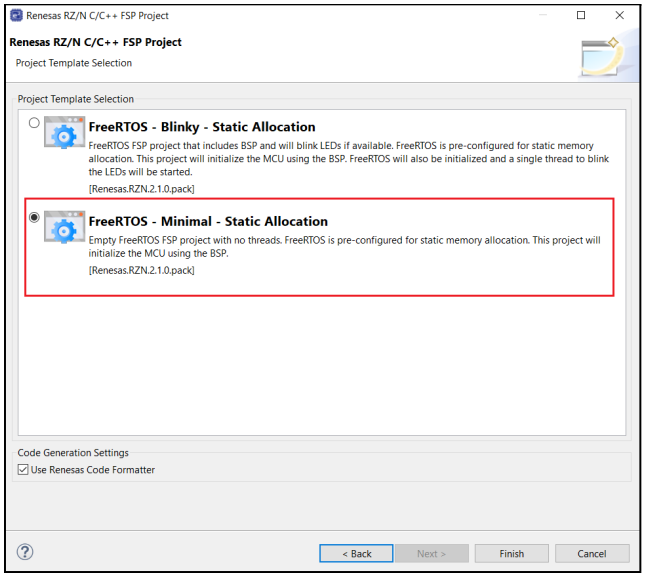


Figure 5-19. Project Template configuration

- Stacks tab: under the Threads section, click New Thread.

- Copy files from the “FreeRTOS” folder to the project.

5.2 Guidelines on EWARM

5.2.1 Baremetal project on IAR EW for Arm

The steps to create a combination project using RZ Smart Configurator are similar to those in e² studio; refer to **Section 5.1 Guidelines on e² studio**.

(1) Build combination

Refer to **4.4.1 Setting for sample program to invocation with I-Jet**.

(2) Configuration the Debug for combination

- The device name format is “Renesas R9A09G087M44_<core>”.
  Where:
  <core>: R52_0 (for CR52_0 core) / R52_1 (for CR52_1 core) / A55 (for CA55_0/1/2/3 core)

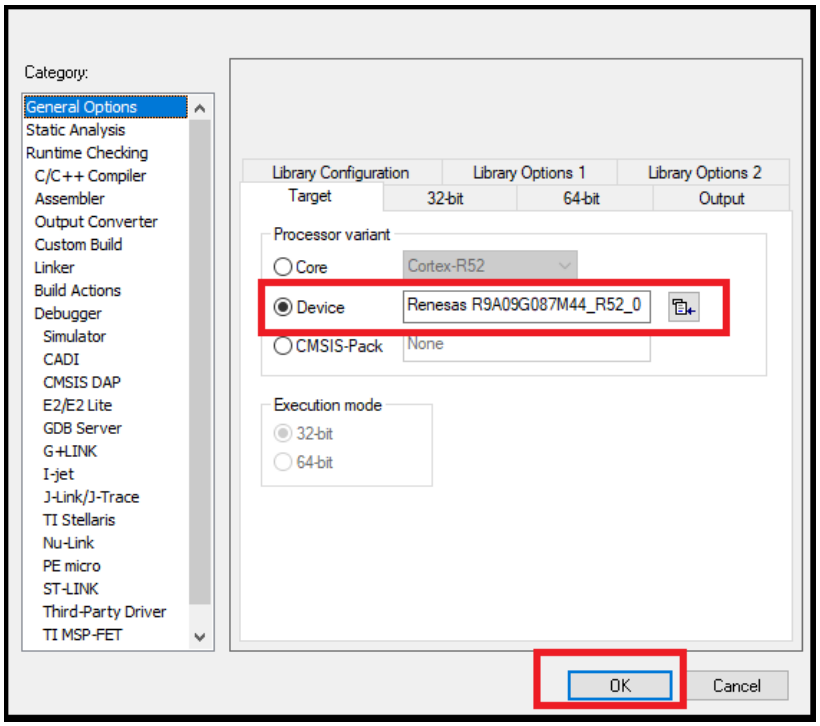


Figure 5-20. Select of device

- Build Actions: Selete “Pre-compile”.

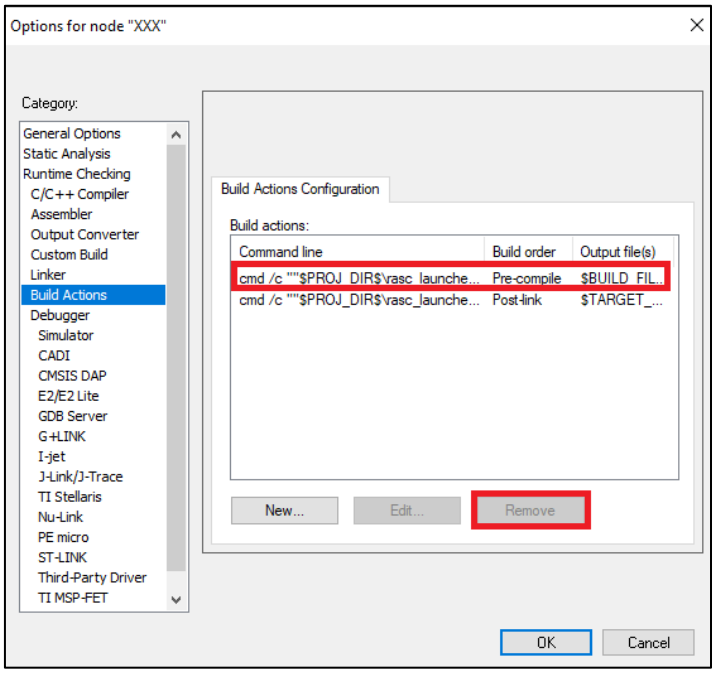


Figure 5-21. Select of device

- Debugger: Select the I-Jet driver.

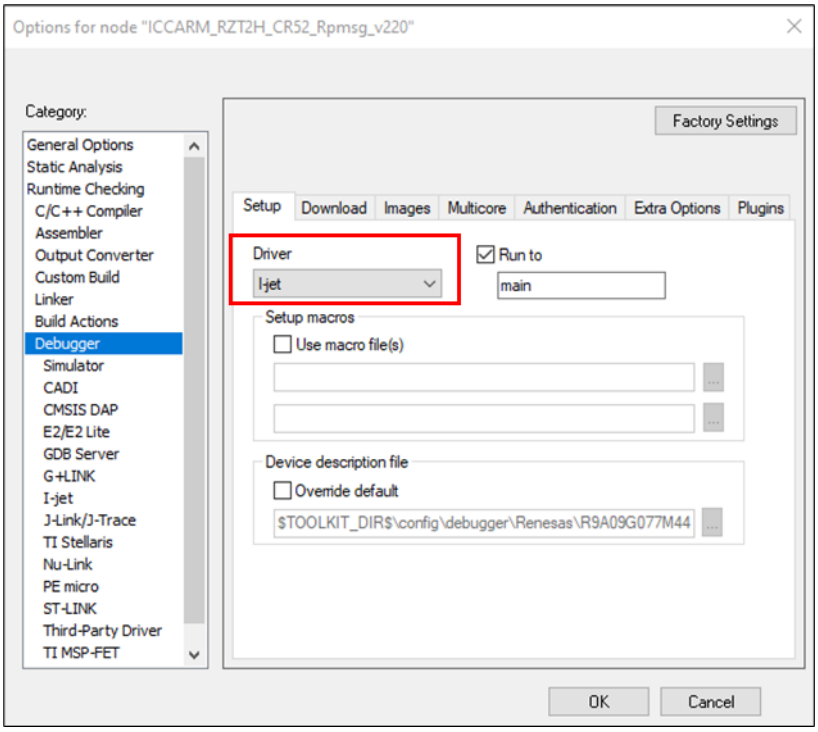


Figure 5-22. Config for debug

Note:

Click Debugger >> Setup, check Use macro file(s) and add "$PROJ_DIR$\script\initialization_TCM.mac" for CR52_1 core.

- **I-Jet:** Select **Hardware (default)** for primary core and **Software** for secondary core.

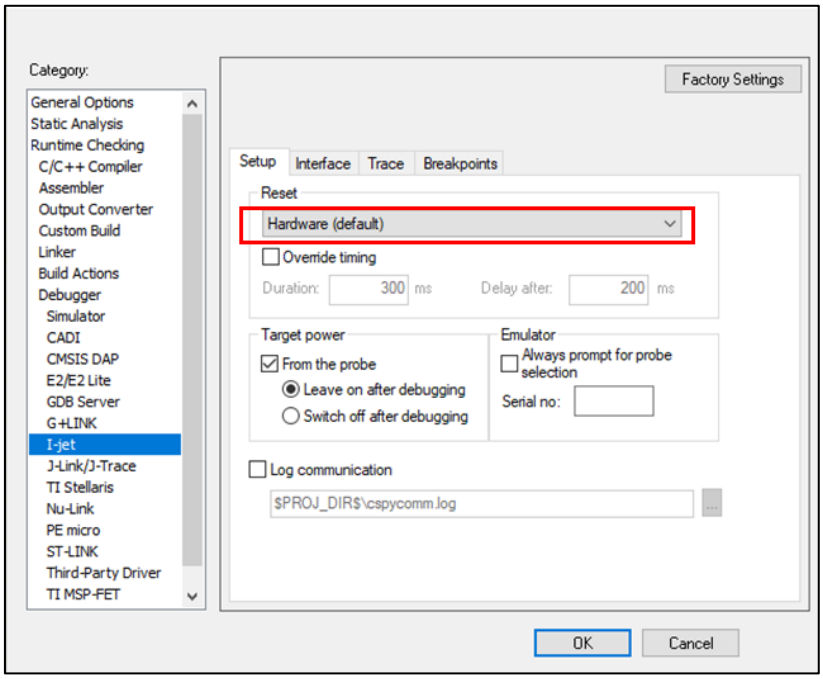


Figure 5-23. Config for debug

- If the project is the primary core, in the Debugger section, under the Multicore tab, select Simple for Asymmetric Multicore if the combination includes 2 projects, and Advanced if the combination includes more than 2 projects.
  - Partner workspace: $PROJ_DIR$\..\[*secondary_core_project*]\[*secondary_core_project*].eww
  - Partner project: [*secondary_core_project_name*]
  - Partner configuration: Debug
  - *multicore_setup.xml* for the combination of 3 projects.

```
<?xml version="1.0" encoding="utf-8"?>
<sessionSetup>
<partner>
<name>Partner0</name>
<workspace>$WS_PATH$</workspace>
<project>$PROJ_PATH$</project>
<config>Debug</config>
<numberOfCores>1</numberOfCores>
</partner>
<partner>
<name>Partner1</name>
<workspace>$PROJ_DIR$\..\[The_name_of_the_second_project ]\[The_name_of_the_second
project].eww</workspace>
<project>[The_name_of_the_second_project ]</project>
<config>Debug</config>
```

```
<numberOfCores>1</numberOfCores>
<attachToRunningTarget>false</attachToRunningTarget>
</partner>
<partner>
<name>Partner2</name>
<workspace>$PROJ_DIR$\..\[The_name_of_the_third_project]\[The_name_of_the_third_project].eww
</workspace>
<project>[The_name_of_the_third_project]</project>
<config>Debug</config>
<numberOfCores>1</numberOfCores>
<attachToRunningTarget>false</attachToRunningTarget>
</partner>
</sessionSetup>
```

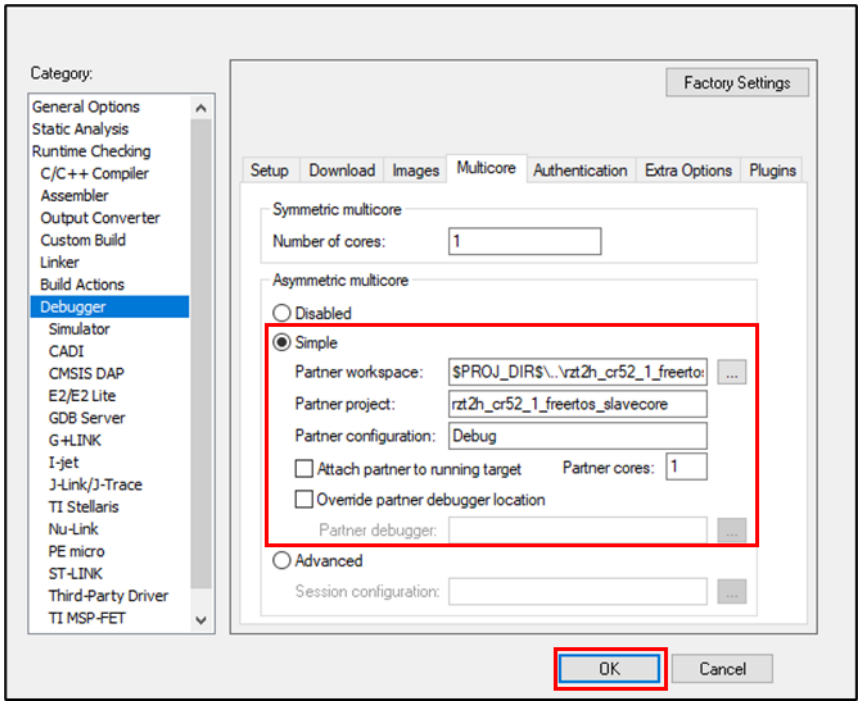


Figure 5-24. Config for debug

- If the project is the secondary core, in the Debugger section, under the Multicore tab, select Disabled for Asymmetric Multicore.

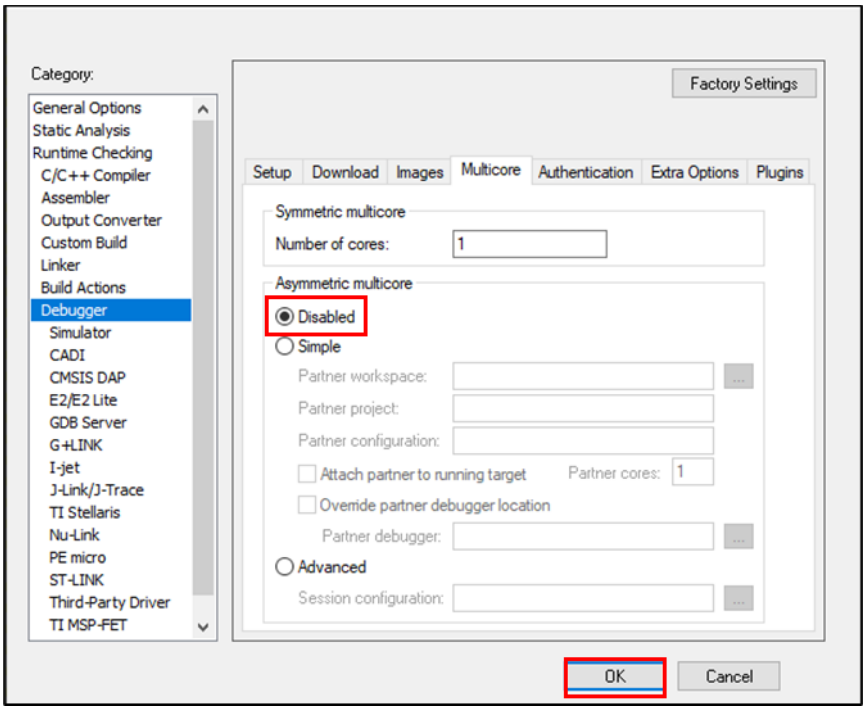


Figure 5-25. Config for debug

After completing the debug configuration, refer to Section 4.5 “Multi-Cores Sample Program Invocation” to run the example.

5.2.2 FreeRTOS project on EWARM

Refer to Section **5.2.1 Baremetal Project on EWARM** to create the project with the following notes:

- Build Artifact and RTOS Selection: Select "FreeRTOS".

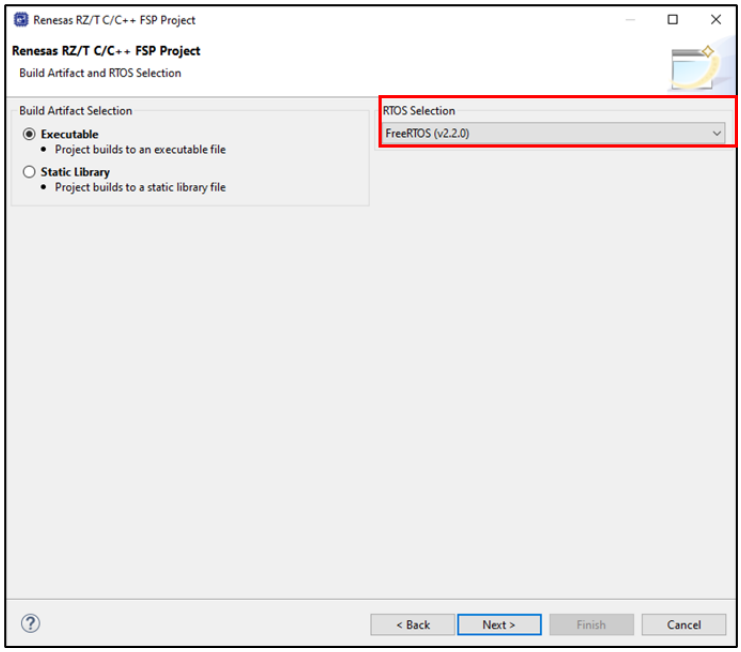


Figure 5-26. Build Artifact and RTOS configuration

- Project Template Selection: Select “FreeRTOS – Minimal – Static Allocation”.

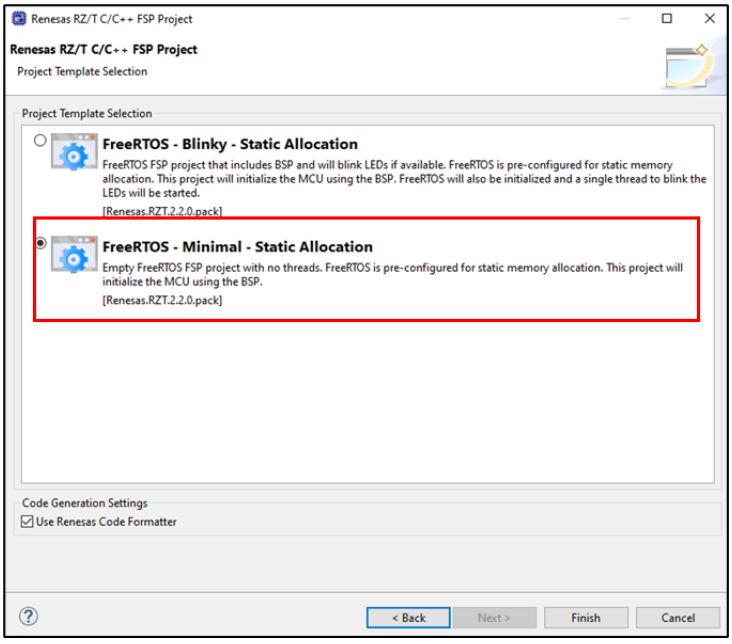


Figure 5-27. Project Template configuration

- Stacks tab: under the Threads section, click New Thread.

- Copy files from the “FreeRTOS” folder to the project

6. Reference documents

• R01AN6434: RZ/T2, RZ/N2 Getting Started with Flexible Software Package

• R01US0821: RZ/T2H and RZ/N2H Evaluation Board - Linux Start-up Guide

Revision History

Rev.	Date	Description	
		Page	Summary
4.00	Mar.31.26	All	First edition issued for the merge of RZ/N2H Multi-OS Quick Start Guide and Application Note

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/.