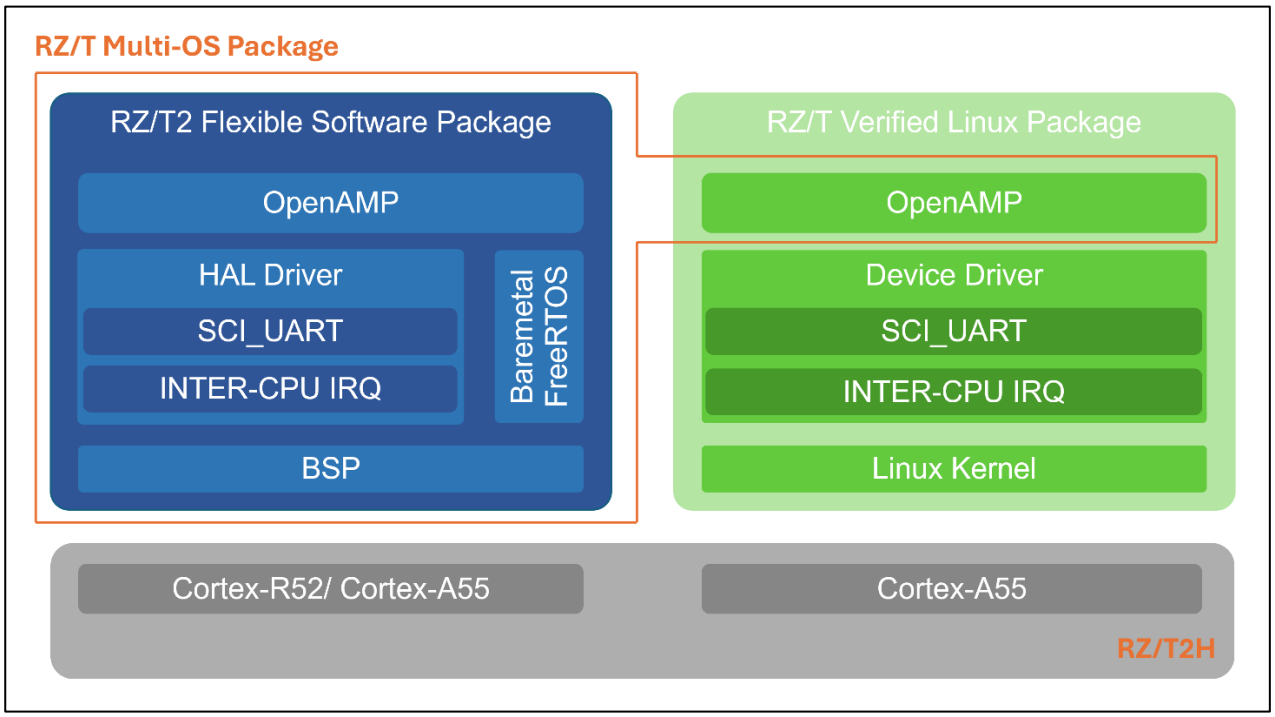
RZ/T2H

RZ/T Multi-OS Package Application Note

Introduction

This software package provides users with easy ways to establish multi-OS (i.e., CIP Linux running on Cortex®-A55 and Bare-metal/ FreeRTOS running on Cortex®-R52/A55) environment and sample program showing how to implement Inter-Processor Communication between those CPU cores.

This package consists of **RZ/T2H** Cortex®-R52/A55 Flexible Software Package (hereinafter referred to as **RZ/T2H** FSP) and Inter-Processor Communication Feature Package for **RZ/T2H** Board Support Package (hereinafter referred to as **RZ/T2H** BSP).



Here are brief descriptions of each component of **RZ/T2H** Multi-OS Package:

- **RZ/T2H FSP**
  The software package consists of production ready peripheral drivers, Bare-metal/ FreeRTOS and portable middleware stacks, and the best in-case HAL drivers with low memory footprint.
- **OpenAMP**
  The framework includes the software components needed for Asymmetric Multiprocessing (AMP) systems such as Inter-Processor Communication.
- **RZ/T2H BSP**
  The environment for building the Board Support Package.

Target device

RZ/T2H

Contents

1. Specifications

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

Table 1-1. Peripheral module to be used

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.

1.1 RZ/T2H reference boards setup

Please refer to Section 3.4 and Section 3.5

1.2 Package information

Table 1-2. Package information

Package name	Usage
RZT_FSP_Packs_3.1.0.zip	RZ/T2 FSP Package
RTK0EF0045Z0042AZJ-v5.0.0.zip	Verified Linux Package
meta-rz-multi-os.tar.gz	RZT2 Multi-OS Layer
gcc_rzt2h_ca55_1_rpmsg_linux_ baremetal _demo.zip	CA55_1 (Baremetal) sample program for e^2 Studio
gcc_rzt2h_ca55_1_rpmsg_linux_freertos_demo.zip	CA55_1 (FreeRTOS) sample program for e^2 Studio
gcc_rzt2h_ca55_2_rpmsg_linux_ baremetal _demo.zip	CA55_2 (Baremetal) sample program for e^2 Studio
gcc_rzt2h_ca55_2_rpmsg_linux_freertos_demo.zip	CA55_2 (FreeRTOS) sample program for e^2 Studio
gcc_rzt2h_ca55_3_rpmsg_linux_ baremetal _demo.zip	CA55_3 (Baremetal) sample program for e^2 Studio
gcc_rzt2h_ca55_3_rpmsg_linux_freertos_demo.zip	CA55_3 (FreeRTOS) sample program for e^2 Studio
gcc_rzt2h_cr52_0_rpmsg_linux_baremetal_demo.zip	CR52_0 (Baremetal) sample program for e^2 Studio
gcc_rzt2h_cr52_0_rpmsg_linux_freertos_demo.zip	CR52_0 (FreeRTOS) sample program for e^2 Studio
gcc_rzt2h_cr52_1_rpmsg_linux_baremetal_demo.zip	CR52_1 (Baremetal) sample program for e^2 Studio
gcc_rzt2h_cr52_1_rpmsg_linux_freertos_demo.zip	CR52_1 (FreeRTOS) sample program for e^2 Studio
iar_rzt2h_ca55_1_rpmsg_linux_ baremetal _demo.zip	CA55_1 (Baremetal) sample program for EWARM
iar_rzt2h_ca55_1_rpmsg_linux_freertos_demo.zip	CA55_1 (FreeRTOS) sample program for EWARM
iar_rzt2h_ca55_2_rpmsg_linux_ baremetal _demo.zip	CA55_2 (Baremetal) sample program for EWARM
iar_rzt2h_ca55_2_rpmsg_linux_freertos_demo.zip	CA55_2 (FreeRTOS) sample program for EWARM
iar_rzt2h_ca55_3_rpmsg_linux_ baremetal _demo.zip	CA55_3 (Baremetal) sample program for EWARM
iar_rzt2h_ca55_3_rpmsg_linux_freertos_demo.zip	CA55_3 (FreeRTOS) sample program for EWARM
iar_rzt2h_cr52_0_rpmsg_linux_baremetal_demo.zip	CR52_0 (Baremetal) sample program for EWARM
iar_rzt2h_cr52_0_rpmsg_linux_freertos_demo.zip	CR52_0 (FreeRTOS) sample program for EWARM
iar_rzt2h_cr52_1_rpmsg_linux_baremetal_demo.zip	CR52_1 (Baremetal) sample program for EWARM
iar_rzt2h_cr52_1_rpmsg_linux_freertos_demo.zip	CR52_1 (FreeRTOS) sample program for EWARM

Table 1-3. Supported boot methods for sample programs

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

Note: The BL2 boot methods for the sample programs of CA55_0 (Linux) and CA55_1, CA55_2, CA55_3 (FSP) are not supported in the RZ/T2H Multi-OS Package v3.00.

2. Verified operation conditions

Table 2-1. Verified operating conditions

Item	Contents	
Microprocessor used	**RZ/T2H**	
Integrated Development Environment	e² Studio 2025-10	IAR EW for Arm (EWARM) 9.60.3
Compiler	**GNU Arm Embedded 13.3.1.arm-13-24** Compiler Options (except directory path): -mcpu=cortex-r52 -mthumb -mfloat-abi=hard -mfpu=neon-fp-armv8 -fdiagnostics-parseable-fixits -Og -fmessage-length=0 -fsigned-char -ffunction-sections -fdata-sections -fno-strict-aliasing -Wunused -Wuninitialized -Wall -Wextra -Wmissing-declarations -Wconversion -Wpointer-arith -Wshadow -Wlogical-op -Waggregate-return -Wfloat-equal -Wnull-dereference -g -o **GCC Arm A-Profile (AArch64 bare-metal) 10.3.1.20210621** Compiler Options (except directory path): -mcpu=generic+simd -mcmodel=small -Og -fmessage-length=0 -fsigned-char -ffunction-sections -fdata-sections -fno-strict-aliasing -Wunused -Wuninitialized -Wall -Wextra -Wmissing-declarations -Wconversion -Wpointer-arith -Wshadow -Wlogical-op -Waggregate-return -Wfloat-equal -g -o	##IAR Ninja build file #Rules rule COMPILER_XCL command = C$:\iar\ewarm-9.60.3\common\bin\XclFileGenerator.exe $xclcommand -f "$rspfile_name" description = IAR_NEW_TOOL+++COMPILER_XCL+++$out rspfile = $rspfile_name rspfile_content = $flags rule INDEXER command = C$:\iar\ewarm-9.60.3\common\bin\SourceIndexer.exe $flags depfile = $out.dep deps = gcc description = IAR_NEW_TOOL+++INDEXER+++$out rule MAKEBROWSE command = C$:\iar\ewarm-9.60.3\common\bin\makeBrowseData.exe $flags description = IAR_NEW_TOOL+++MAKEBROWSE+++$out rule PDBLINK command = C$:\iar\ewarm-9.60.3\common\bin\PbdLink.exe $flags description = IAR_NEW_TOOL+++PDBLINK+++$out

3. Sample program setup

3.1 RZ/T2H Flexible Software Package setup

Please refer to **RZ/T2H** Getting Started with Flexible Software Package.

3.2 OpenAMP related stuff integration to RZ/T2H Board Support Package

This section describes how to integrate OpenAMP related stuff to **RZ/T2H** Verified Linux Package. For details on how to build the VLP, please refer RZ/T2H and RZ/N2H Evaluation Board - Linux Start-up Guide.

(1) Install the package needed for building BSP

Before starting the build, run the command below on the Linux Host PC to install packages used for building the BSP.

```
$ sudo apt-get update
$ sudo apt-get install build-essential chrpath cpio debianutils diffstat file \
gawk gcc git iputils-ping libacl1 liblz4-tool libssl-dev libyaml-dev \
locales python3 python3-git python3-jinja2 python3-pexpect python3-pip \
python3-subunit socat texinfo tmux unzip wget xterm xz-utils zstd \
bmap-tools libsdl1.2-dev p7zip-full fdisk tmux
```

(2) Configure your Git

Run the commands below and set the user name and email address before starting the build procedure. Without this setting, an error occurs when building procedure runs git command to apply patch.

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

(3) Create a working directory at your home directory, and decompress Yocto recipe package

```
$ mkdir ~/rzt2h_bsp_${PACKAGE_VERSION}
$ export WORK=~/rzt2h_bsp_${PACKAGE_VERSION}
$ cd $WORK
$ cp <path to folder download package> \
/RTK0EF0045Z0042AZJ-${PACKAGE_VERSION}.zip .
$ cp <path to folder download package>/ meta-rz-multi-os.tar.gz .
$ unzip ./RTK0EF0045Z0042AZJ-${PACKAGE_VERSION}.zip
$ tar zxvf ./RTK0EF0045Z0042AZJ-${PACKAGE_VERSION} \
/rzt2h_n2h_bsp_${PACKAGE_VERSION}.tar.gz
$ tar zxvf ./meta-rz-multi-os.tar.gz
```

Note: Please note that your build environment must have 100GB of free hard drive space in order to complete the minimum build. The Yocto BSP build environment is very large. Especially in case you are using a Virtual Machine, please check how much disk space you have allocated for your virtual environment.

(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. After building the Linux BSP, you must re-prepare the Micro-SD card (core-image-minimal-rzt2h-dev.rootfs.wic.gz) and rewrite the bootloader files (bl2_bp_xspi0-rzt2h-dev.srec and fip-rzt2h-dev.srec) to the target board. Details on preparing the Micro-SD card and writing the bootloader files are described in sections 3.3 Preparing the SD Card and 3.4.2 Booting Flash Writer.

(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. After building the Linux BSP, you mSust rewrite the bootloader files (bl2_bp_xspi0-rzt2h-dev.srec and fip-rzt2h-dev.srec) to the target board. Detailed instructions for writing the bootloader files are provided in section 3.4.2 Booting Flash Writer.

(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 = " RZT2H_CA55_CR52_BL2_BOOT"
```

And, select target core.

Ex: When CR52_0 is the slave core.

```
MACHINE_FEATURES_append = " RZT2H_CR520_BL2_BOOT"
```

Note: After building the Linux BSP, you must rewrite the bootloader files (bl2_bp_xspi0-rzt2h-dev.srec and fip-rzt2h-dev.srec) to the target board. Only one target core should be selected at a time. Detailed instructions for writing the bootloader files are provided in section 3.4.2 Booting Flash Writer.

(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
```

Note: After building the Linux BSP, you must re-prepare the Micro-SD card (core-image-minimal-rzt2h-dev.rootfs.wic.gz). Detailed instructions for preparing the Micro-SD card can be found in 3.3 Preparing the SD Card.

(8) Build Initialize

Initialize a build using the 'oe-init-build-env' script in Poky and point TEMPLATECONF to platform conf path.

```
$ TEMPLATECONF=$PWD/meta-renesas/meta-rz-distro/conf/templates/vlp-v5-conf/ \
source poky/oe-init-build-env build
```

(9) Add the necessary layers for building a multi-OS

Please run the command below to add the multi-os layer.

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

(10) Start a build

Run the commands below to start a build. Building an image can take up to a few hours depending on the user's host system performance. Build the target file system image using bitbake.

```
$ MACHINE=rzt2h-dev bitbake core-image-minimal
```

After the build is successfully completed, a similar output will be seen, and the command prompt will return.

```
NOTE: Tasks Summary: Attempted 7427 tasks of which 16 didn't need to be rerun
and all succeeded.
```

All necessary files are located in the 'build/tmp/deploy/images' folder.

3.3 Preparing the SD Card

To boot from Micro-SD card, over 4GB capacity of blank Micro-SD card is needed. You can use Linux Host PC to expand the kernel and the rootfs using USB card reader or other equipment.

(1) Set Micro-SD card to Linux PC. And check the mount device name with fdisk command

```
$ sudo fdisk -l
Disk /dev/sdb: 3.74 GiB, 3997171712 bytes, 7806976 sectors
Disk model: Storage Device
Units: sectors of 1 * 512 = 512 bytes
Disklabel type: dos
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 $
$ umount /dev/sdb1
$ umount /dev/sdb2
```

If necessary, replace “/dev/sdb” with the correct device name for your system.

(2) Expand disk image

```
$ cd $WORK/build/tmp/deploy/images/rzt2h-dev
$ sudo bmaptool copy core-image-minimal-rzt2h-dev.rootfs.wic.gz /dev/sdb
```

When you complete the above commands, the Micro-SD card is prepared correctly.

Note: When switching the macros ENABLE_REMOTEPROC and RPMSG_REMOTE_CORE, the user need to repeat this step.

3.4 Bootloader for RZ/T2H

Refer to section 4 of the Linux Start-up Guide for further instructions.

3.4.1 Setting boot mode to SCI (UART) boot mode

Connect between the board and a control PC by USB serial cable.

(1) Bring up the terminal software and select the “File” >> “New Connection” to set the connection on the software

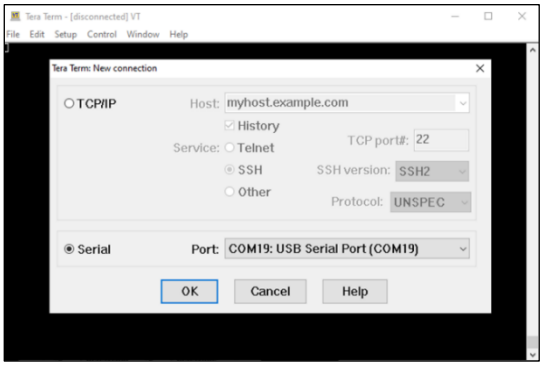


Figure 3.1 Set “Serial”

(2) Select the “Setup” >> “Serial port” to set the settings about serial communication protocol on the software

- Set the settings about serial communication protocol on a terminal software as below.

- Speed: 115200 bps
- Data: 8bit
- Parity: None
- Stop bit: 1bit
- Flow control: None
- Transmission delay 0ms/line (L)

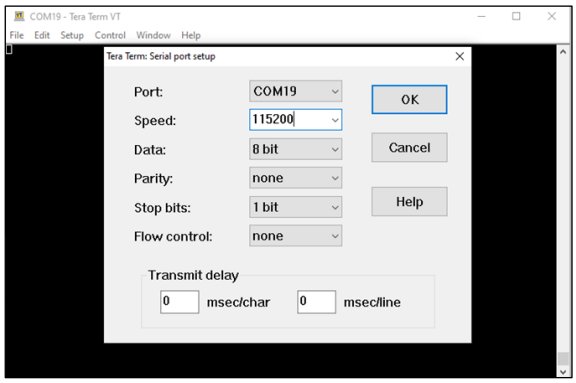


Figure 3.2 Terminal Setting

(3) To set the board to SCI (UART) Boot mode, set the SW14 as below

Table 3.1. Mode Setting Switch SW14

Boot mode	SCI (UART) Boot mode
SW14-1	OFF
SW14-2	ON
SW14-3	OFF
SW14-4	OFF
SW14-5	OFF
SW14-6	ON
SW14-7	ON
SW14-8	OFF

(4) After finished all settings, when pressed the reset button SW16, the messages below are displayed on the terminal

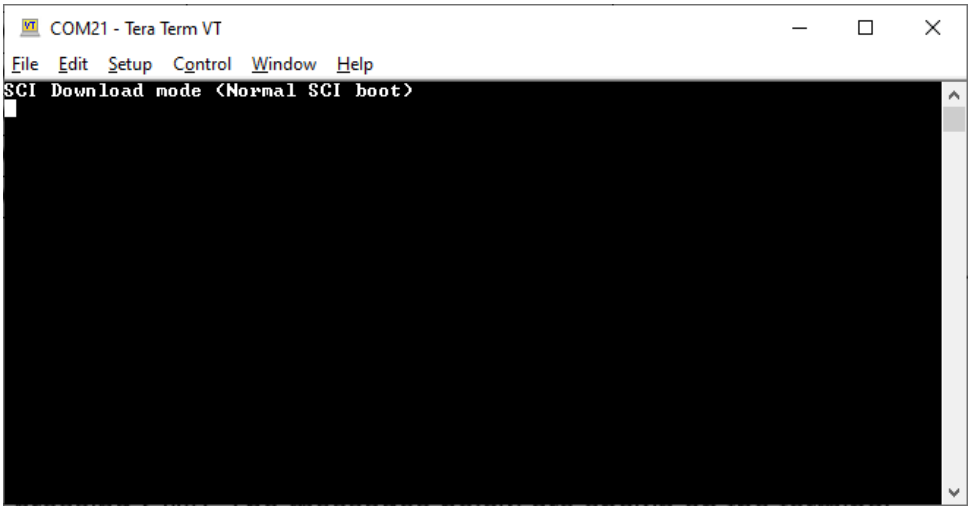


Figure 3.3 Terminal log of SCI (UART) download mode

3.4.2 Booting flash writer

Turn on the power of the board by pressing SW16. The messages below are shown on the terminal.

```
SCI Download mode (Normal SCI boot)
```

(1) Send 2 images of Flash Programmer using terminal software after the message “please send !” is shown. Send “**HDR NM**” firstly, then send “**Flash_Programmer_SCIF_CR52_RZT2H_EVK.mot**” secondary.

(2) Open a “**Send file**” dialog by selecting “**File**” → “**Send file**” menu.

(3) Send file > “**HDR NM**”

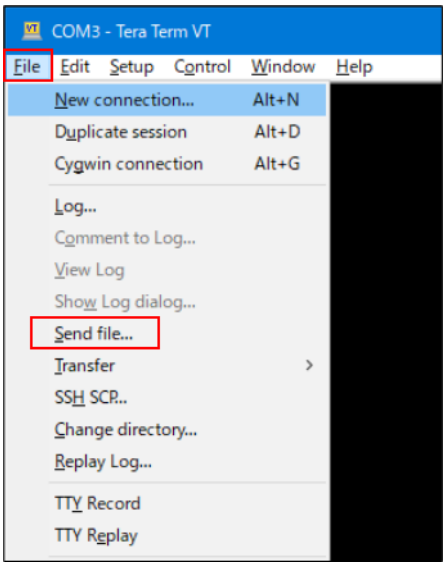


Figure 3.4 Select “Send file”

(4) The image will be sent to the board via serial connection. After completing that, send “Flash_Programmer_SCIF_CR52_RZT2H_EVK.mot” as same as “HDR NM”. Send it as soon as possible.

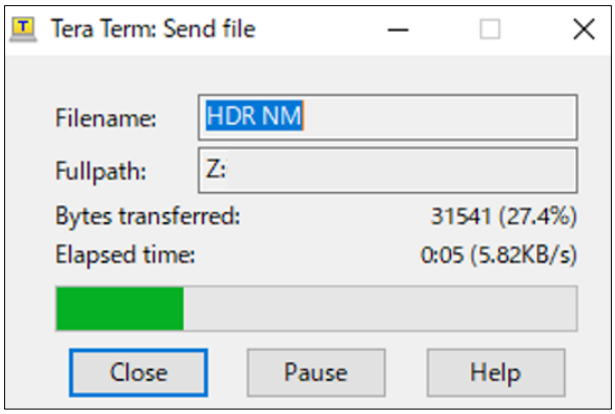


Figure 3.5 Sending File

(5) When the transfer is complete, please transfer the file of “Flash_Programmer_SCIF_CR52_RZT2H_EVK.mot” using the same procedure within 10 seconds. If not completed within 10 seconds, please restart the transfer from the “HDR NM" file.

After successfully downloading the binary, Flash Writer starts automatically and shows a message like below on the terminal.

```
SCI Download mode (Normal SCI boot)
-- Load Program to RAM  ---------------
-- Start Boot Program on RAM ---------
(Flash Programmer V1.04 (RZT2H)(CR52_0) (built: Oct 9 2024, 21:59:10)
```

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

Send file > "bl2_bp_xspi0-rzt2h-dev.srec"

```
Erased
Writen
```

```
> XSPIW 0 0x060000 0          # input command
send file
```

Send file > "fip-rzt2h-dev.srec"

```
Erased
Writen
```

After writing two loader files normally, turn off the power of the board by changing the POWER_SW.

Note: When switching the macros ENABLE_REMOTEPROC, ENABLE_U_BOOT, MACHINE_FEATURES _append, the user need to repeat this step.

3.5 Setting U-boot

To set the board to xSPI0 Boot mode, set the SW14 as below:

Boot mode	xSPI0 boot mode
SW14-1	ON
SW14-2	ON
SW14-3	ON
SW14-4	OFF
SW14-5	OFF
SW14-6	OFF
SW14-7	ON
SW14-8	OFF

To connect Micro-SD Card Slot:

Switch	Settings of Configuration Circuits
SW2-3	ON
SW5-3	OFF
SW5-4	ON

Turn on the power of the board by pressing the reset button SW16.

```
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/T2H
Model: Renesas Development EVK based on r9a09g077m44
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 ##(Press enter before the countdown end)
=>
```

Following the messages above, many warning messages will be shown. These warnings are eliminated by setting correct environment variables. Please set value and save them to the Flash ROM.

```
=> env default -a
## Resetting to default environment
=> saveenv
Saving Environment to MMC... Writing to MMC(0)....OK
=> boot
```

4. Sample program invocation

4.1 Hardware setup

(1) Connect J-Link OB to RZ/T2H

(2) Connect the USB to Serial Port to the RZ/T2H

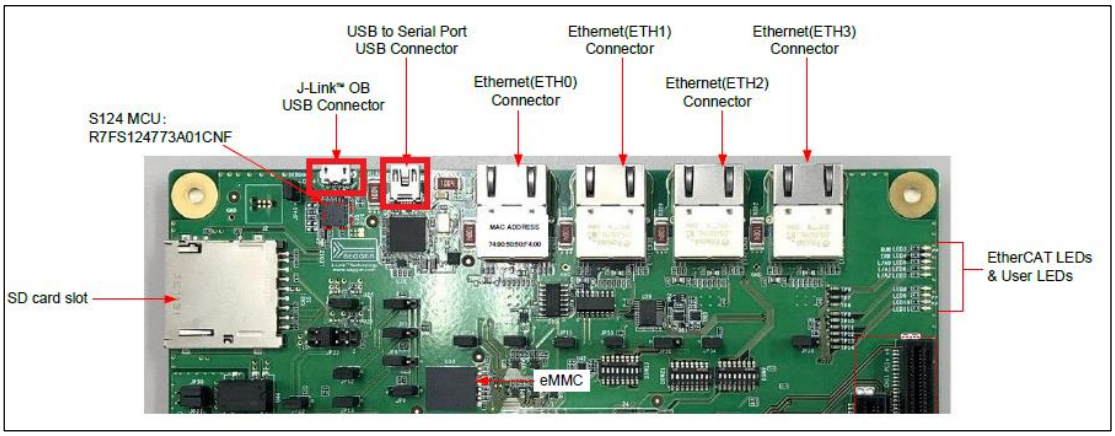


Figure 4-1. Connection J-Link, SCI (UART) to RZ/T2H board

4.2 Sample program setup on e² Studio

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

(1) Extract gcc_<device>_rpmsg_linux_<platform_type>_demo.zip there.

<device>: rzt2h_cr52_0/ rzt2h_cr52_1/ rzt2h_ca55_1/ rzt2h_ca55_2/ rzt2h_ca55_3

<platform_type>: baremetal/ freertos

(2) Open e² Studio and click **File** >> **Import**.

(3) Double-click General and select **Existing Projects into Workspace**.

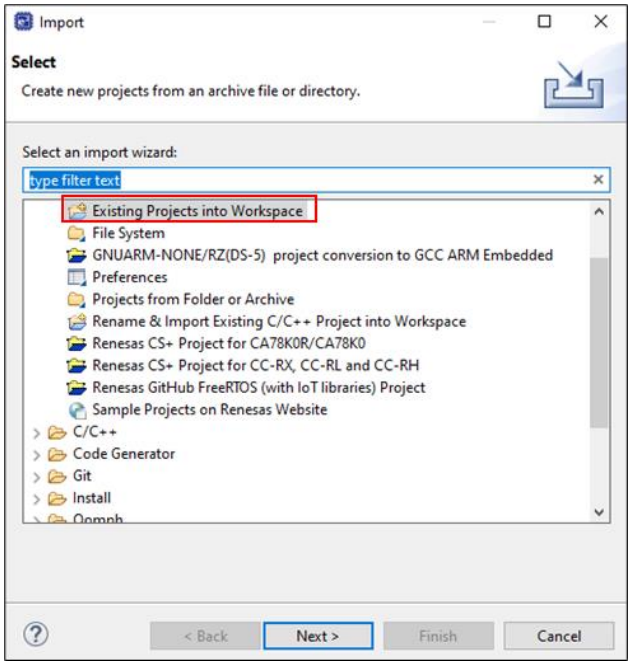


Figure 4-2. Import sample project on e² Studio (1)

(4) Input the path to the folder gcc_<device>_rpmsg_linux_<platform_type>_demo, press Enter key and click Finish button.

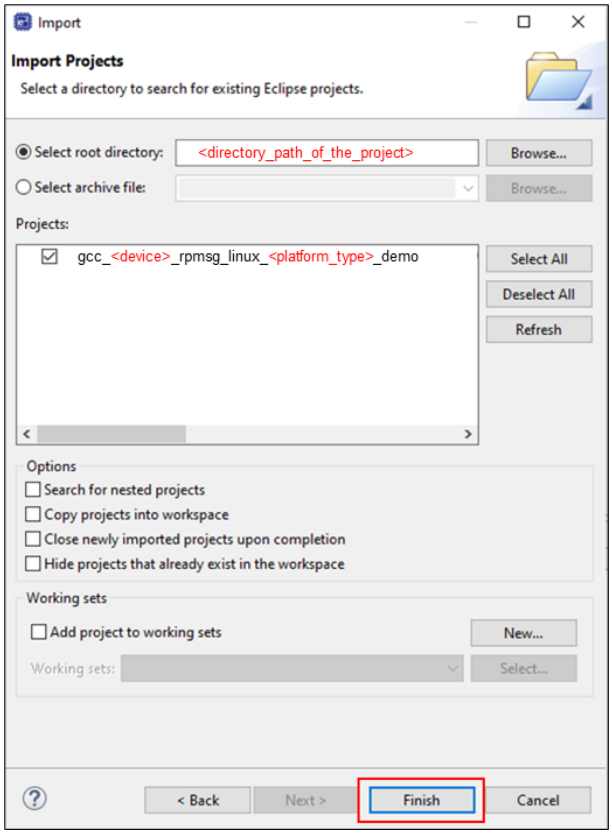


Figure 4-3. Import sample project on e² Studio (2)

4.2.1 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.

4.2.2 Setting for sample program to invocation with remoteproc

When invoking with remoteproc, perform the following step:

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

Core	Current	Modify
CR52_0/ CR52_1	#define ENABLE_REMOTEPROC (0U)	#define ENABLE_REMOTEPROC (1U)

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

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.

4.2.3 Setting for sample program to invocation with u-boot

When invoking with u-boot, perform the following step:

(1) In the BSP tab configuration, navigate to RZT Common >> Port Protect and set it to Disabled.

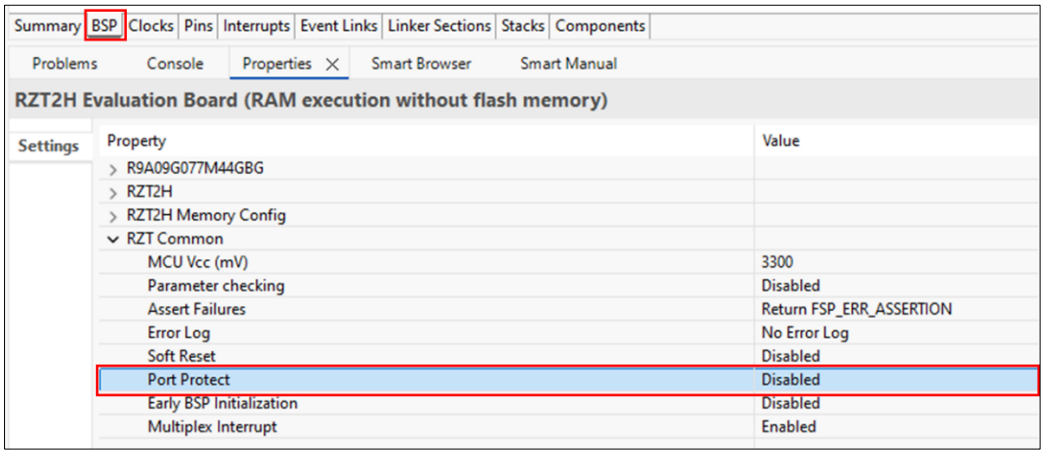


Figure 4-4. BSP tab configuration

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

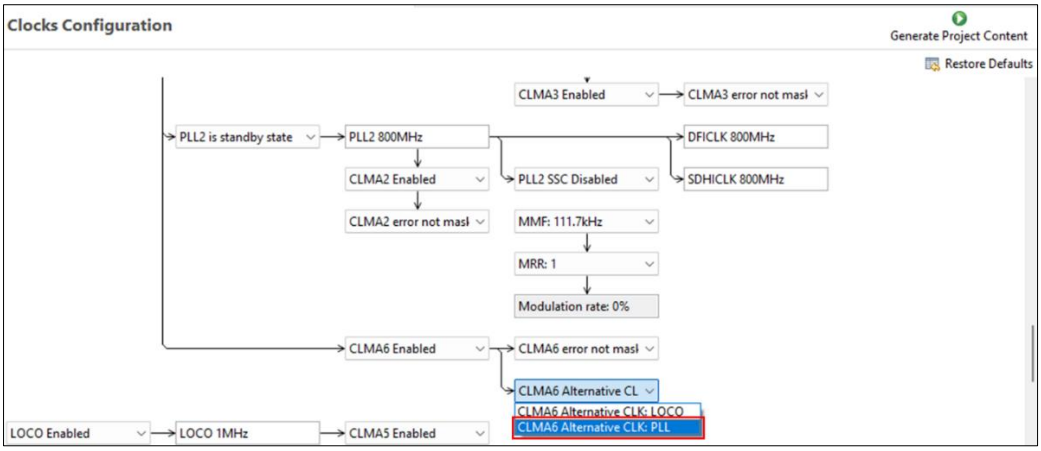


Figure 4-5. Clocks tab configuration

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

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

(4) 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.

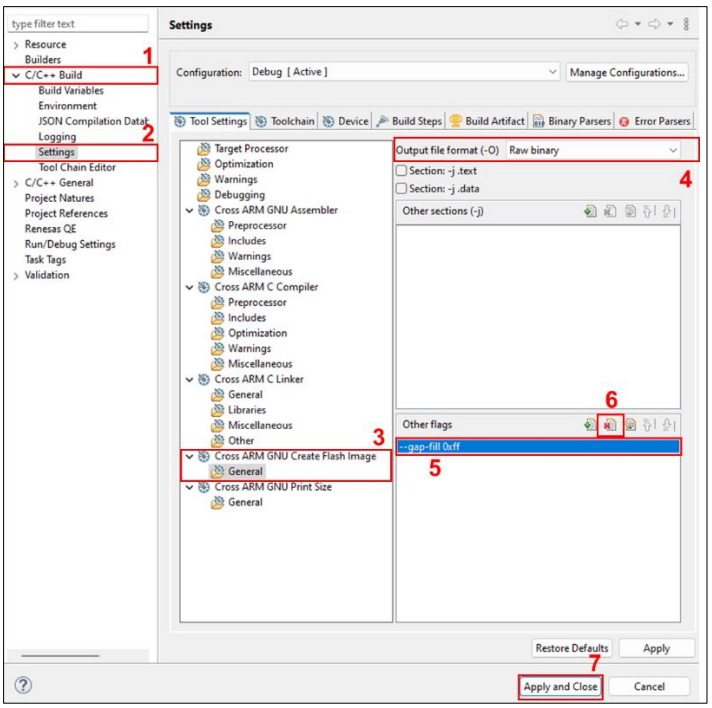


Figure 4-6. Firmware Binary Configuration on e2 Studio

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

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

When invoking with BL2 of Trusted Firmware-A, 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) In the BSP tab configuration, navigate to RZT Common >> Port Protect and set it to Disabled.

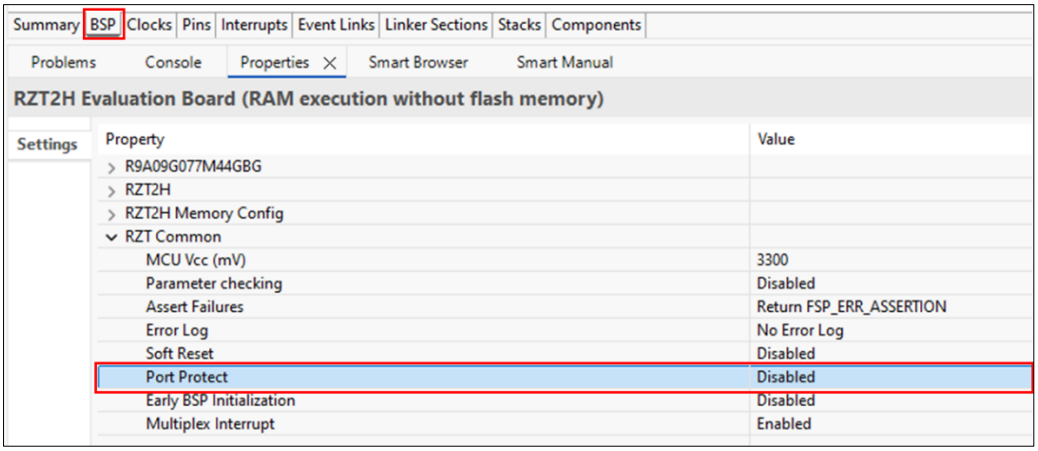


Figure 4-7. BSP tab configuration

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

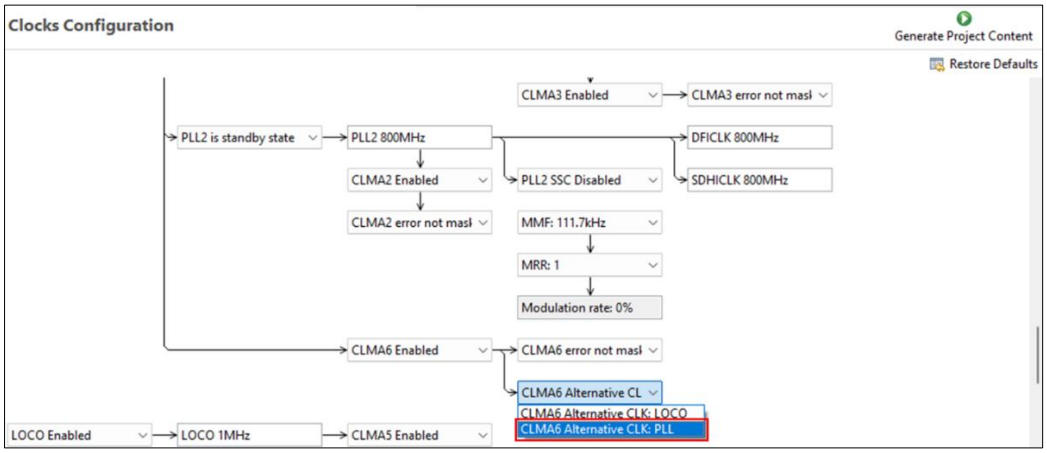


Figure 4-8. Clocks tab configuration

(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.

4.3 Sample program set up on EWARM

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

(1) Extract iar_<device>_rpmsg_linux_<platform_type>_demo.zip there.

<device>: rzt2h_cr52_0/ rzt2h_cr52_1/ rzt2h_ca55_1/ rzt2h_ca55_2/ rzt2h_ca55_3

<platform_type>: baremetal/ freertos

(2) Open EWARM and click File >> Open Worksapce.

(3) Navigate to the folder containing the project extracted in step 1, select the file iar_<device>_rpmsg_linux_<platform_type>_demo.eww, then click Open.

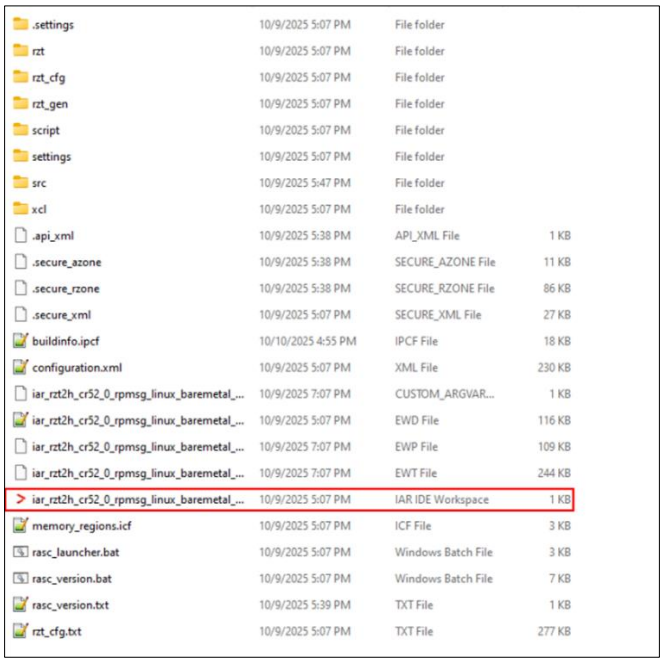


Figure 4-9. Import sample project on EWARM

4.3.1 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.

4.3.2 Setting for sample program to invocation with remotepoc

When invoking with remoteproc, perform the following step:

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

Core	Current	Modify
CR52_0/ CR52_1	#define ENABLE_REMOTEPROC (0U)	#define ENABLE_REMOTEPROC (1U)

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

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.

4.3.3 Setting for sample program to invocation with u-boot

When invoking with u-boot, perform the following step:

(1) In the BSP tab configuration, navigate to RZT Common >> Port Protect and set it to Disabled.

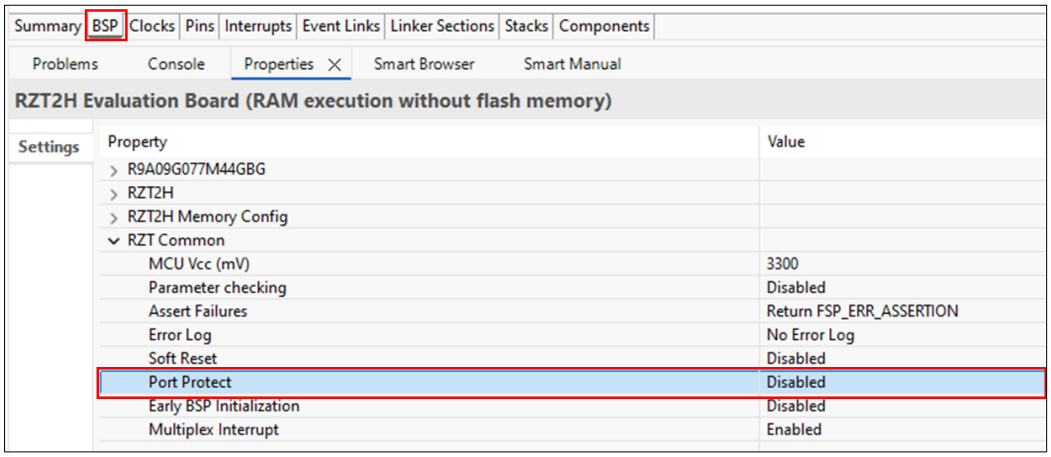


Figure 4-10. BSP tab configuration

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

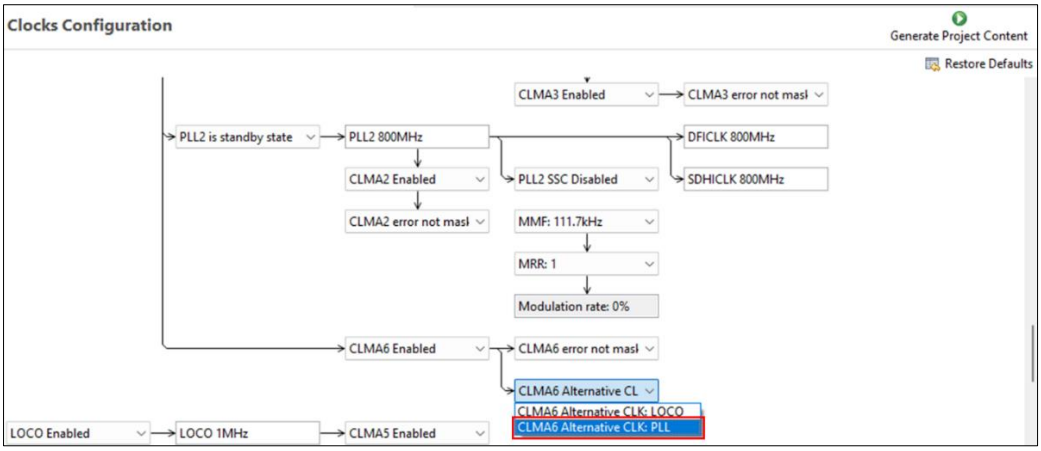


Figure 4-11. Clocks tab configuration

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

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

(4) Modify the buildinfo.ipcf file by moving the files listed below from the “Component” group to another group. (For example, it can be moved to the “Program Entry” group, similar to the sample program)

```
<path>rzt/linaro/libmetal/lib/device.c</path>
<path>rzt/linaro/libmetal/lib/dma.c</path>
<path>rzt/linaro/libmetal/lib/init.c</path>
<path>rzt/linaro/libmetal/lib/io.c</path>
<path>rzt/linaro/libmetal/lib/log.c</path>
<path>rzt/linaro/libmetal/lib/shmem.c</path>
```

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

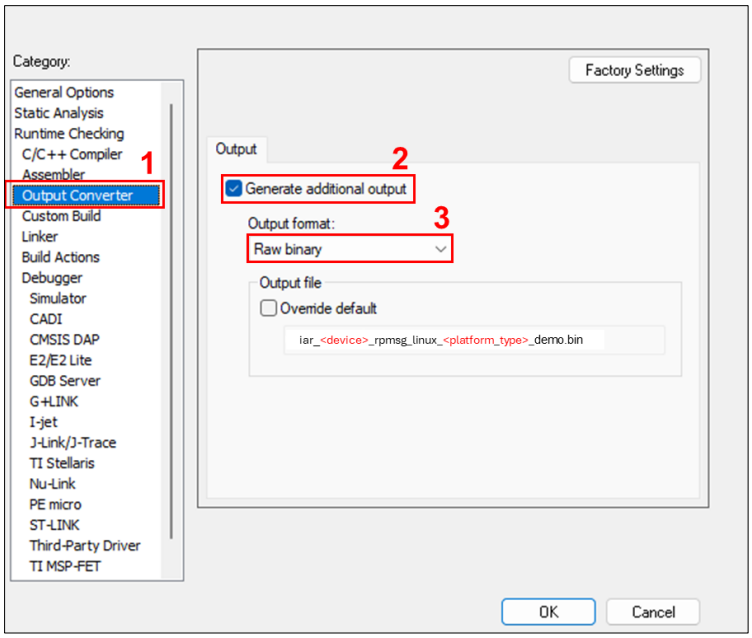


Figure 4-12. Firmware Binary Configuration on EWARM

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

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

When invoking with BL2 of Trusted Firmware-A, 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) In the BSP tab configuration, navigate to RZT Common >> Port Protect and set it to Disabled.

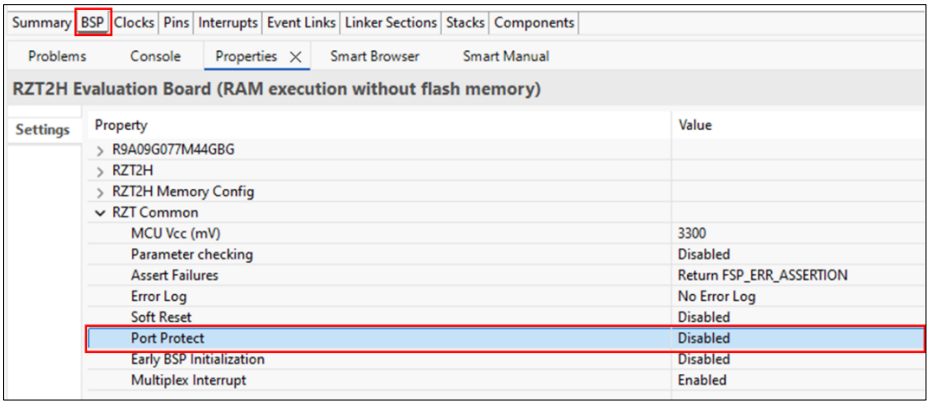


Figure 4-13. BSP tab configuration

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

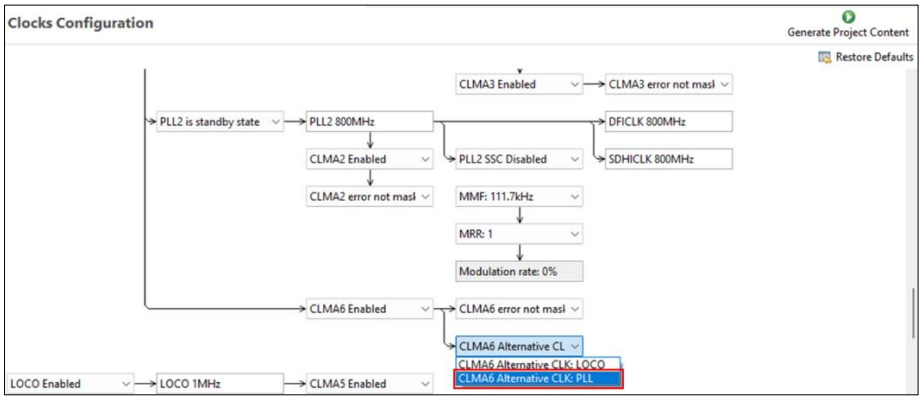


Figure 4-14. Clocks tab configuration

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

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

(6) The .srec file can be found in the Debug folder. However, EWARM only generates .srec files in the S1 record format. 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
```

4.4 Multi-cores sample program invocation

4.4.1 Multi-Cores sample program invocation with Segger J-Link on e² Studio

The procedure of this sample program follows steps to invoke Multi-Cores sample program with Segger J-Link on e² Studio.

(1) Log in as “root” on the Linux terminal

```
rzt2h-dev login: root
```

(2) Configure the debugger

(1) Clicking **Run** >> **Debugger Configurations** or by selecting the dropdown menu next to the bug icon and selecting Debugger Configurations.

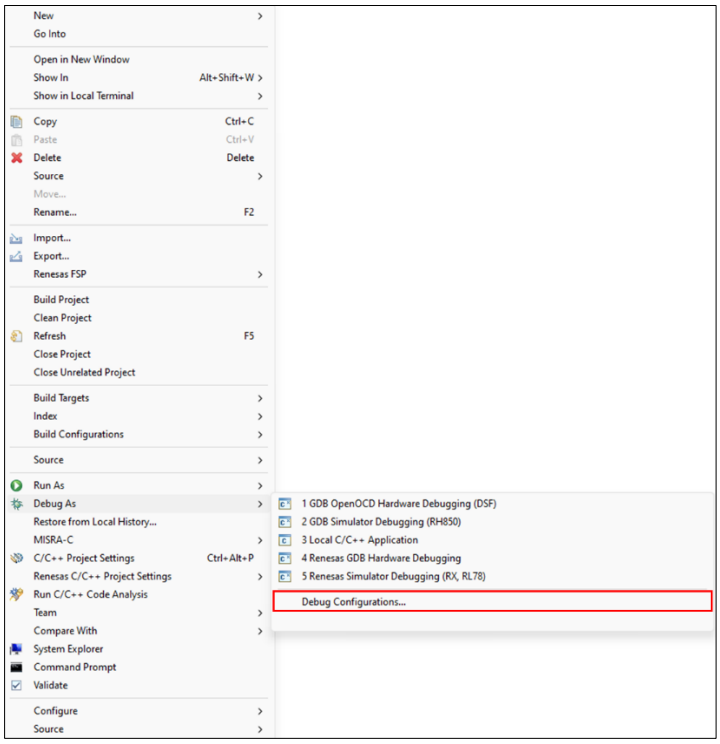


Figure 4-15. Select of Debug Configuration

Note: For the sample program of the CR52_1/ CA55_1/ CA55_2/ CA55_3 cores, the rzt2h_cr52_0_boot project only needs to complete the project build process and does not need to be downloaded to the board.

(2) In the **Debug Configurations** window, select an existing debug configuration or create a new one by double-clicking **Renesas GDB Hardware Debugging**.

- If you select an existing debug configuration, click on the project and then press the **Debug** button to complete.

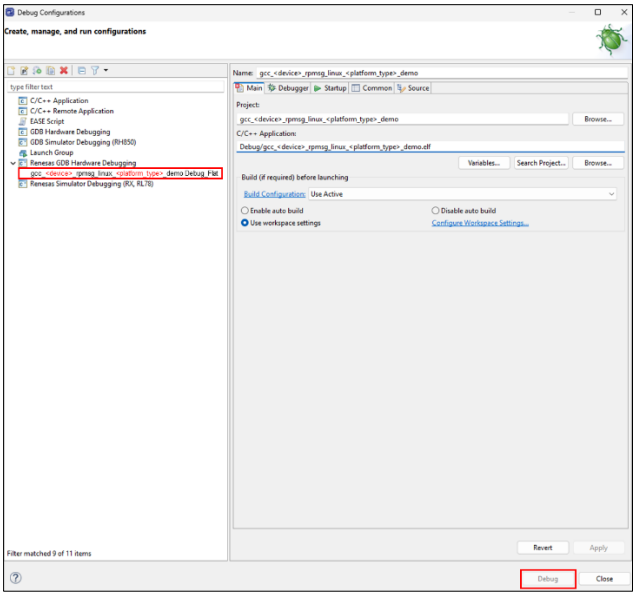


Figure 4-16. Debug Configuration (1)

- If you create a new debug configuration, please set up the debug settings as shown below.

Main tab:

- Set gcc_<device>_rpmsg_linux_<platform_type>_demo to **Project with Browser** button.

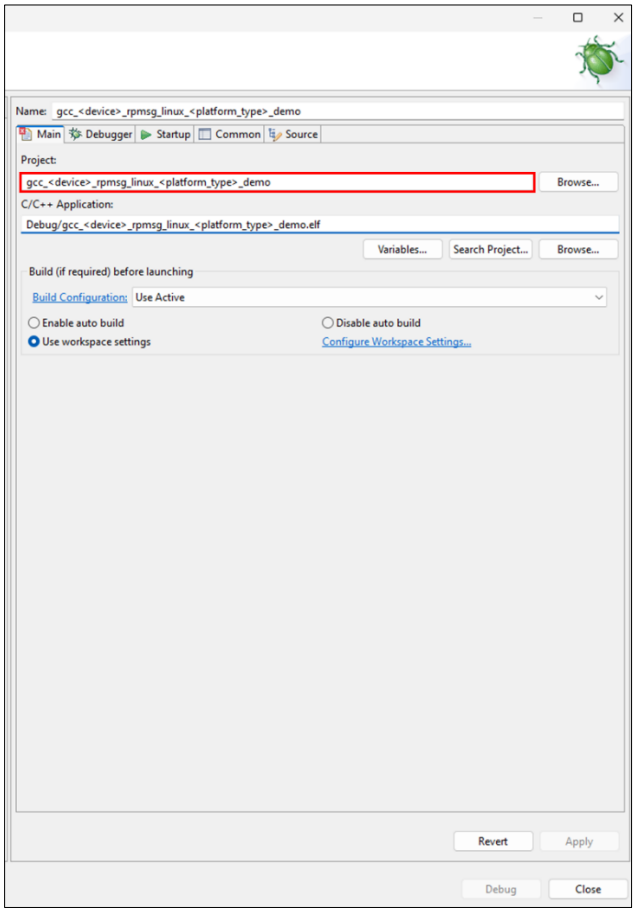


Figure 4-17. Debug Configuration (2)

Debugger tab >> GDB Settings:

- Debug hardware select **J-Link ARM**.
- Target Device select **RZ** >> **RZ/RZT2H** >> **R9A07G077M44**_<core>.
  <core>: CR52_0/ CR52_1/ CA55_1/ CA55_2/ CA55_3

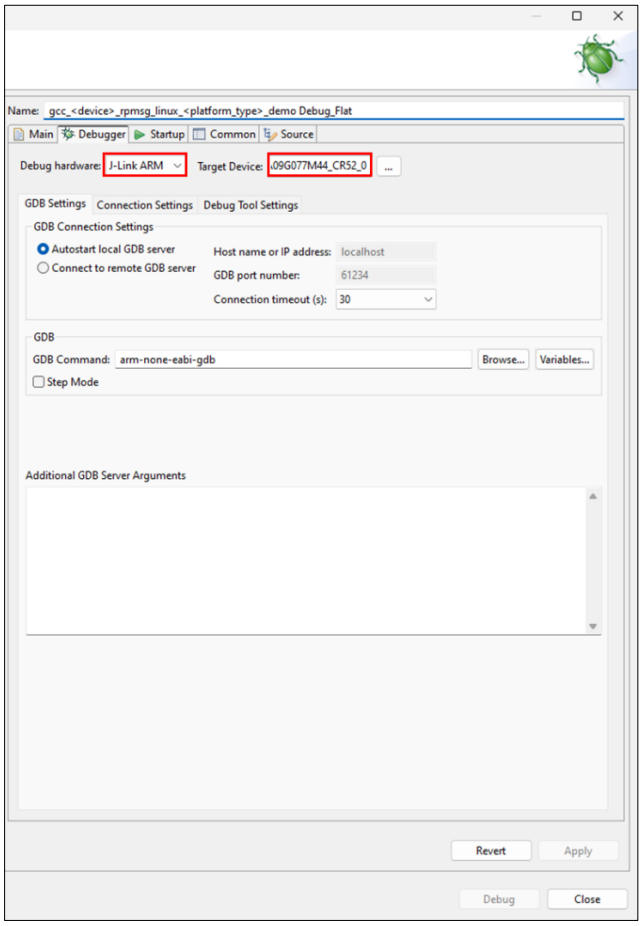


Figure 4-18. Debug Configuration (3)

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).

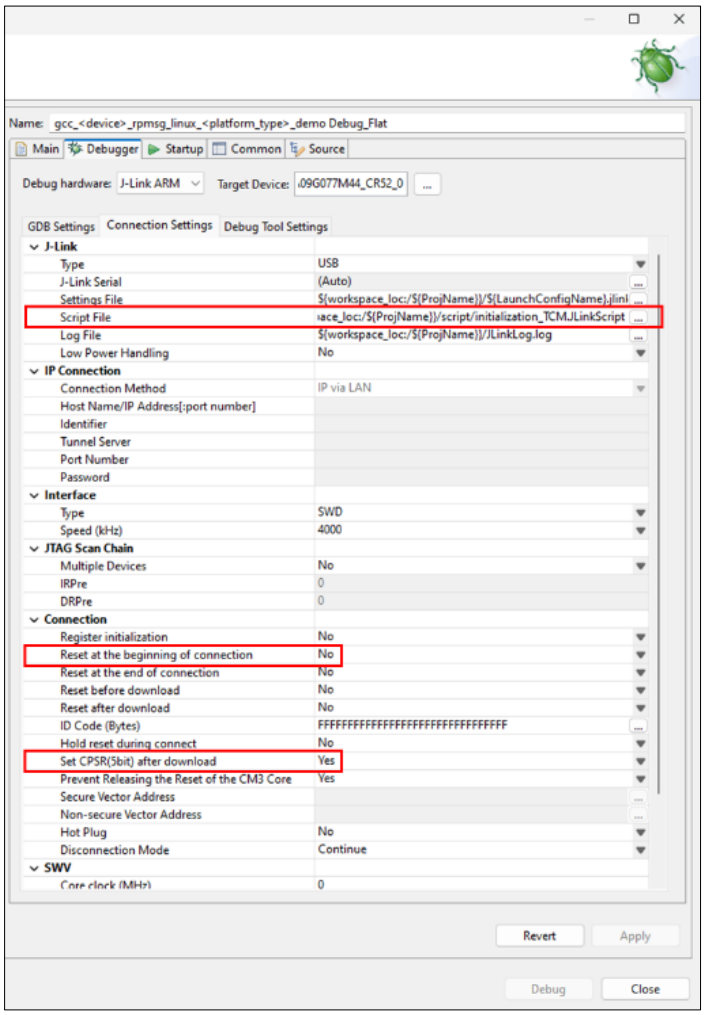


Figure 4-19. Debug Configuration (4)

(3) Click Apply then Debug button

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

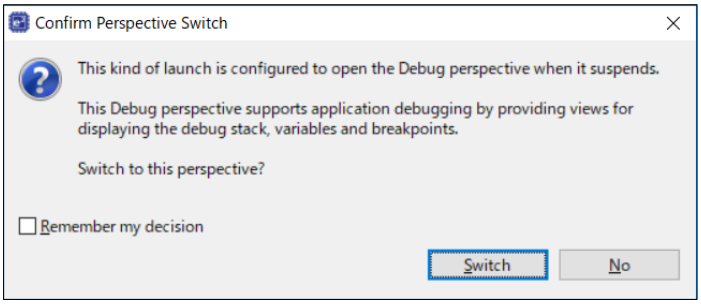


Figure 4-20. Confirmation window to open the Debug perspective

(4) Run the Multi-cores Project

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

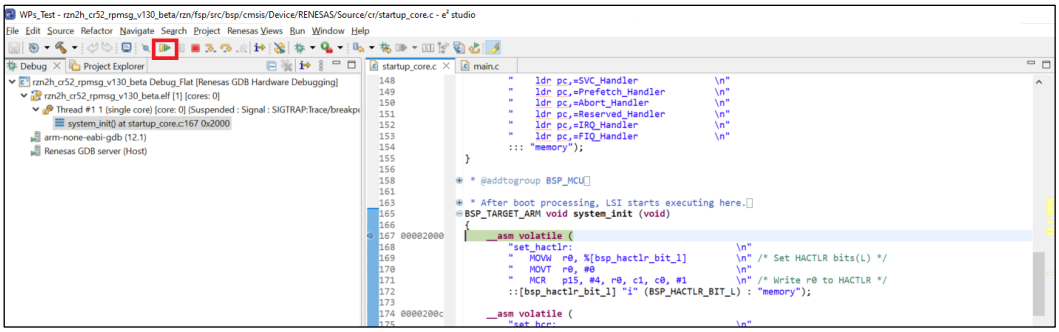


Figure 4-21. e² Studio Debugger Memory Window

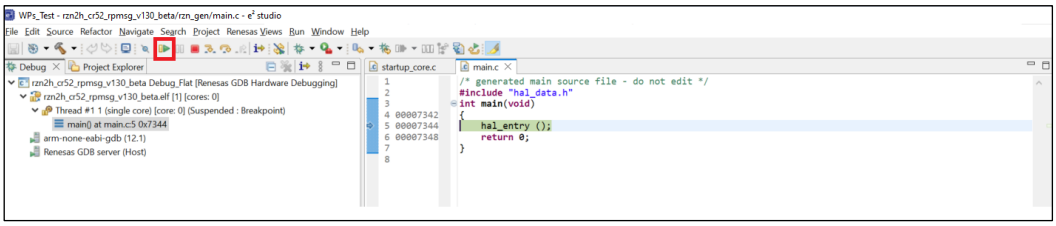


Figure 4-22. Hal entry

(5) Now that FSP sample program has started, the following message is shown on the console connected to SCI (UART)

```
***************  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
```

At this point of time, FSP program is waiting for the establishment of rpmsg channel between FSP and Linux.

(6) Run Linux sample program by executing the following command on Linux

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

```
root@rzt2h-dev:~# rpmsg_sample_client 0
```

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

(1) 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
NOTICE:  BL31: Built : 17:00:45, Feb 28 2025
U-Boot 2021.10 (Mar 13 2025 - 09:27:21 +0000)
CPU: Renesas Electronics RZ/T2H
Model: Renesas Development EVK based on r9a09g077m44
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
=>
```

(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/r9a09g077m44-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)

(3) Execute the command below.

```
root@rzt2h-dev:~# rpmsg_sample_client 0
```

(7) 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@rzt2h-dev:~#
```

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

The procedure of this sample program follows steps to invoke multi-cores sample program with I-Jet on EWARM.

(1) Log in as “root” on the Linux terminal

```
rzt2h-dev login: root
```

(2) Configure the debugger for the project

- Clicking **Options** >> **General Options**. In the **Target** tab, under **Processor variant**, check if the **Device** field has the correct device name. If not, adjust it accordingly.
- The device name format is “Renesas R9A09G077M44_<core>”.
  <core>: R52_0 (for CR52_0 core)/ R52_1 (for CR52_1 core)/ A55 (for CA55_1/2/3 core)

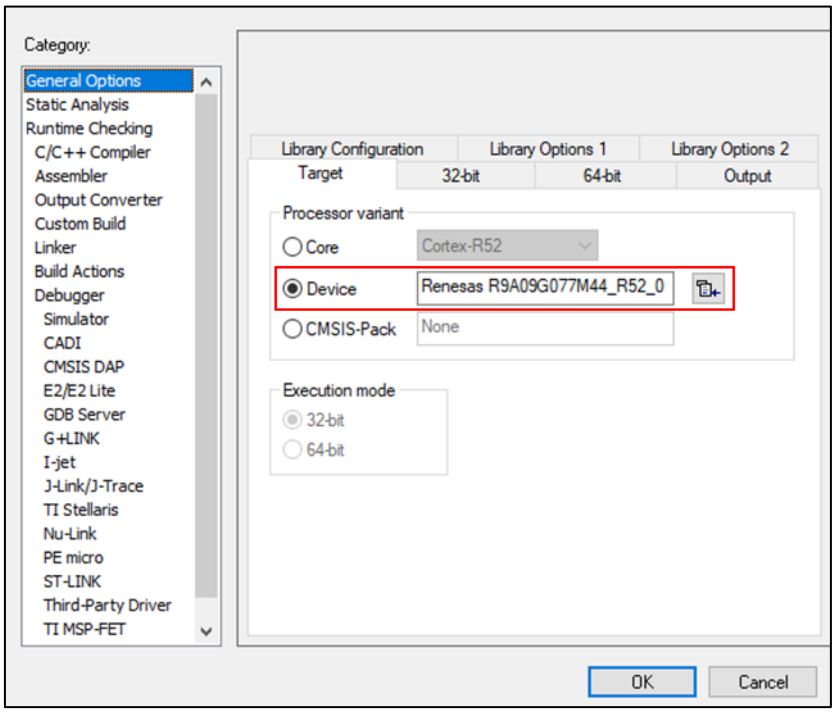


Figure 4-23. Select of Options

Note: For the sample program of the CR52_1/ CA55_1/ CA55_2/ CA55_3 cores, the rzt2h_cr52_0_boot project only needs to complete the project build process and does not need to be downloaded to the board.

- In the **Debugger** section, under the **Setup** tab, select the **I-Jet** driver.

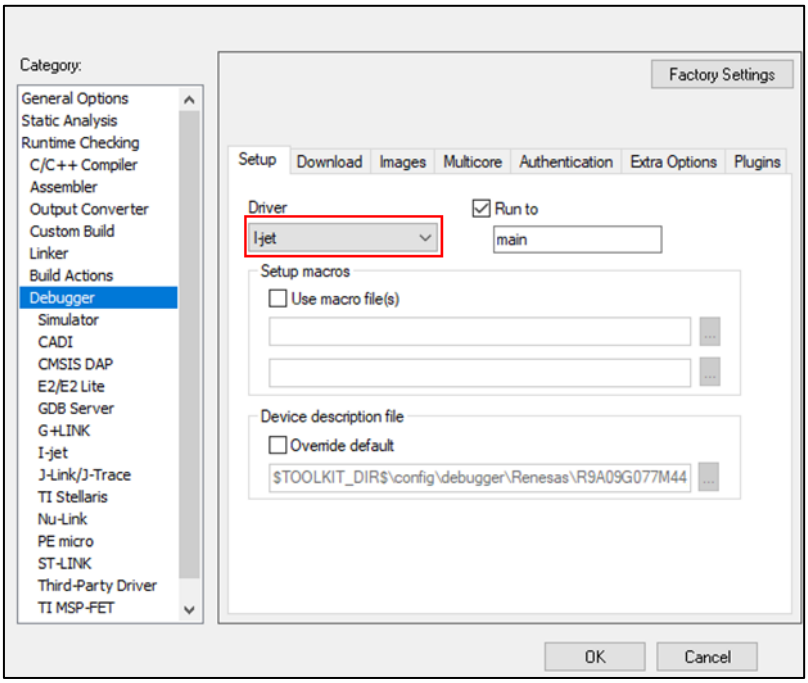


Figure 4-24. Select of Options

- In the **I-Jet** section, under the **Setup** tab, select **Reset Disabled (no reset)**. Click the "**OK**" button to complete the setup

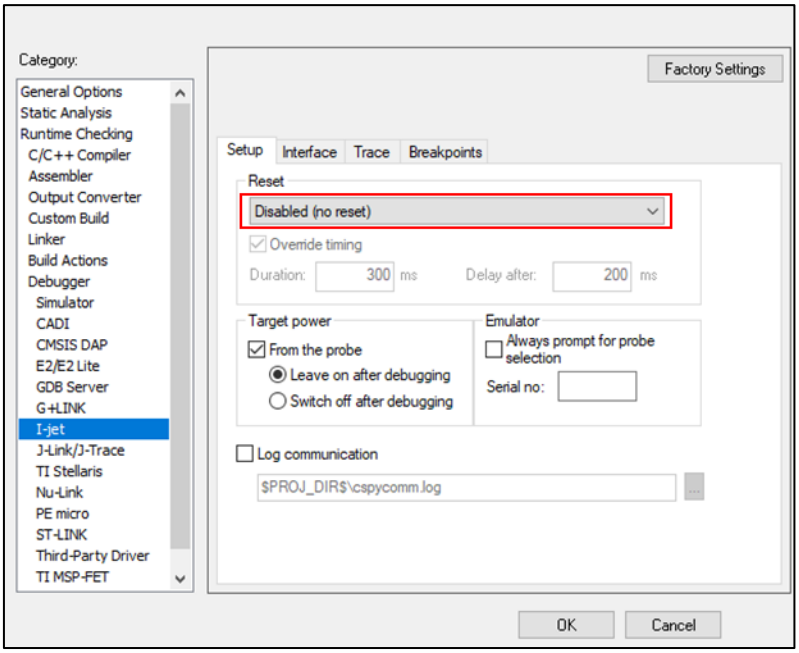


Figure 4-25. Select of Options

(3) Run the multi-cores project

- Click the **Download and Debug** button on the toolbar to enter debug mode.

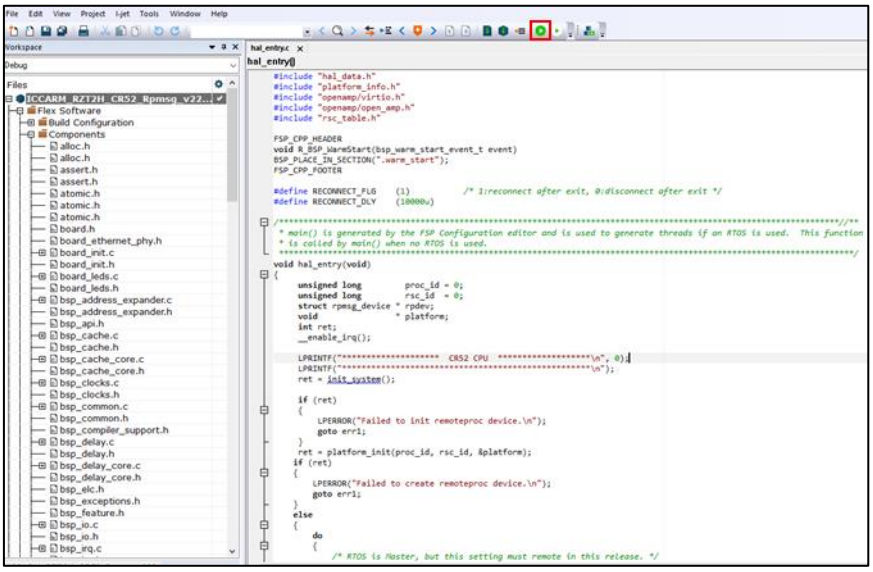


Figure 4-26. Download and Debug

- While in **Debug** mode, click the **Go** icon on the toolbar.

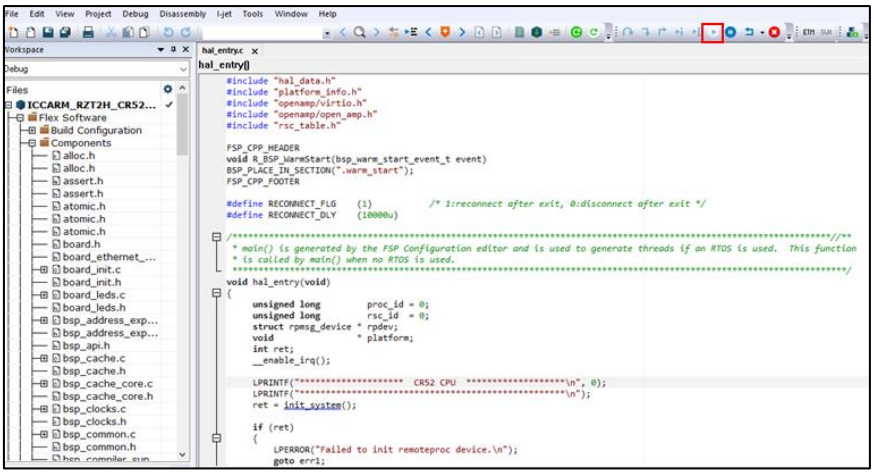


Figure 4-27. Download and Debug

(4) Now that FSP sample program has been started, the following message is shown on the console connected to SCI (UART)

```
***************  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
```

At this point of time, FSP program is waiting for the establishment of rpmsg channel between FSP and Linux.

(5) Run Linux sample program by executing the following command on Linux

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

```
root@rzt2h-dev:~# rpmsg_sample_client 0
```

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

(1) 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
NOTICE:  BL31: Built : 17:00:45, Feb 28 2025
U-Boot 2021.10 (Mar 13 2025 - 09:27:21 +0000)
CPU: Renesas Electronics RZ/T2H
Model: Renesas Development EVK based on r9a09g077m44
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
=>
```

(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/r9a09g077m44-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)

(3) Execute the command below.

```
root@rzt2h-dev:~# rpmsg_sample_client 0
```

(6) 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 : 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@rzt2h-dev:~#
```

4.4.3 Multi-cores sample program invocation with remoteproc

On **RZ/T2H** evaluation board, you can invoke RPMsg sample program with remoteproc by following the procedure stated below:

(1) Booting up Linux by following "5. Booting and Running Linux" of RZ/T2H and RZ/N2H Evaluation Board Linux Start-up Guide

Note: Re-preparing the Micro-SD card (core-image-minimal-rzt2h-dev.rootfs.wic.gz) and rewriting the bootloader files (bl2_bp_xspi0-rzt2h-dev.srec and fip-rzt2h-dev.srec) to the target board are required after building the Linux BSP with ENABLE_REMOTEPROC ?= "1". Details on preparing the Micro-SD card and writing the bootloader files are described in sections 3.3 Preparing the SD Card and 3.4.2 Booting Flash Writer.

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

```
root@rzt2h-dev:~# echo
<env>_<device>_rpmsg_linux_<platform_type>_demo.<extension>
> /sys/class/remoteproc/<remoteprocX>/firmware
```

<env>: gcc/ iar.

<device>: rzt2h_cr52_0/ rzt2h_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" folder located at "~/rzt2h_bsp_v1.0.2/build" and rebuild the Linux BSP from scratch.

(3) Kick CR52 by using the command below

```
root@rzt2h-dev:~# echo start > /sys/class/remoteproc/remoteprocX/state
```

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

```
root@rzt2h-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@rzt2h-dev:~# rpmsg_sample_client 0
```

(4) Stop CR52 by using the command below

```
root@rzt2h-dev:~# echo stop > /sys/class/remoteproc/remoteprocX/state
```

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

```
root@rzt2h-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@rzt2h-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@rzt2h-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-rzt2h-dev.wic.gz.

```
$ MACHINE=rzt2h-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 unmount /media/
```

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

4.4.4 Multi-cores sample program invocation with u-boot

On **RZ/T2H** 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

(1) 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/
```

(2) 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>: rzt2h_cr52_0/ rzt2h_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/T2H evaluation board

(3) Turn on the RZ/T2H 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/T2H
Model: Renesas Development EVK based on r9a09g077m44
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>: rzt2h_cr52_0/ rzt2h_cr52_1.

<platform_type>: baremetal.

<core>: cr520/ cr521.

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

```
root@rzt2h-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@rzt2h-dev:~# shutdown -h now
```

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

On **RZ/T2H** 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-rzt2h-dev.srec and fip-rzt2h-dev.srec) as described in section 3.4.2 Booting Flash Writer, 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>: rzt2h_cr52_0/ rzt2h_cr52_1.

<platform_type>: baremetal.

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

(2) Refer to section 3.5 Setting U-Boot to boot the board.

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

4.5 Overview of sample program behavior

The behavior of sample program is as follows:

(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/T2H 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/T2H 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: GIC (Generic Interrupt Controller) 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.

5. Reference documents

- R01AN6983: RZ/T2 Getting Started with Flexible Software Package
- R01US0681: RZ/T2H and RZ/N2H Board Support Package
- R01US0682: RZ/T2H and RZ/N2H Evaluation Board - Linux Start-up Guide

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Nov.22.24	All	• First edition issued
1.10	Dec.04.24	All	• Updated document name
1.11	Mar.10.25	3, 5 6, 27, 28 16, 22	• Update Linux BSP Package 1.0.1 • Support remoteproc boot (GCC) • Add a note when executing the CR52_1 project.
1.20	Apr.28.25	3, 5 27 11 3, 7, 13, 15, 18, 20, 22, 25, 26	• Update Linux BSP Package 1.0.2 • Support remoteproc boot (IAR) • Update switch to connect Micro-SD Card Slot • Support combinations for FSP (CA55_1/2/3) – Linux.
1.30	Oct.17.25	3 3 37 7, 17, 21, 41 7, 18, 22, 43	• Update FSP Package v3.0.0 • Support combinations for FSP (CA55_1/2/3 Baremetal) – Linux • Support remoteproc boot (TCM region) • Support u-boot boot • Support BL2 of TF-A boot
3.00	Dec.25.25	-	• Update FSP Package v3.1.0 • Update Linux Verify Package v5.0.0 • Unsupported BL2 boot sample programs on CA55

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/.