RZ/G2L, RZ/G2LC and RZ/G2UL-EVKIT

R01US0798EJ0101
Rev.1.01
Nov 30, 2025

Linux Start-up Guide

Introduction

This document provides a guide to prepare RZ/G2L, RZ/G2LC and RZ/G2UL reference boards to boot up with the Verified Linux Package.

This guide provides the following information:

- Building procedure
- Preparation for use
- Boot loader and U-Boot
- How to run this Linux package on the target board
- How to create a software development kit (SDK)

Target Reference Board

The target boards of this VLP are as below table.

Table 1. Target boards list

Device	Evaluation Board
RZ/G2L	RZ/G2L Evaluation Board Kit (P/N: RTK9744L23S01000BE)
RZ/G2LC	RZ/G2LC Evaluation Board Kit (P/N: RTK9744C22S01000BE)
RZ/G2UL	RZ/G2UL Evaluation Board Kit (P/N: RTK9743U11S01000BE)

🚧 Please refer to Appendix 8.2 and 8.3 for how to distinguish and replace each board.
🚧 Please refer to Appendix 8.4 for how to connect the Parallel to HDMI Conversion board for RZ/G2UL Evaluation board Kit (smarc-rzg2ul).

Target Software

- RZ MPU Verified Linux Package version 4.0.1 or later. (hereinafter referred to as “VLP”)

Contents

1. Environmental Requirement

The environment for building the Verified Linux Package (hereinafter referred to as “VLP”) is listed in **Table 2**. Please refer to the documents below for details about setting up the environment:

Figure 1 shows the recommended environment for this package.

A Linux PC is required for building the software.

A Windows PC can be used as a serial terminal interface with software such as TeraTerm.

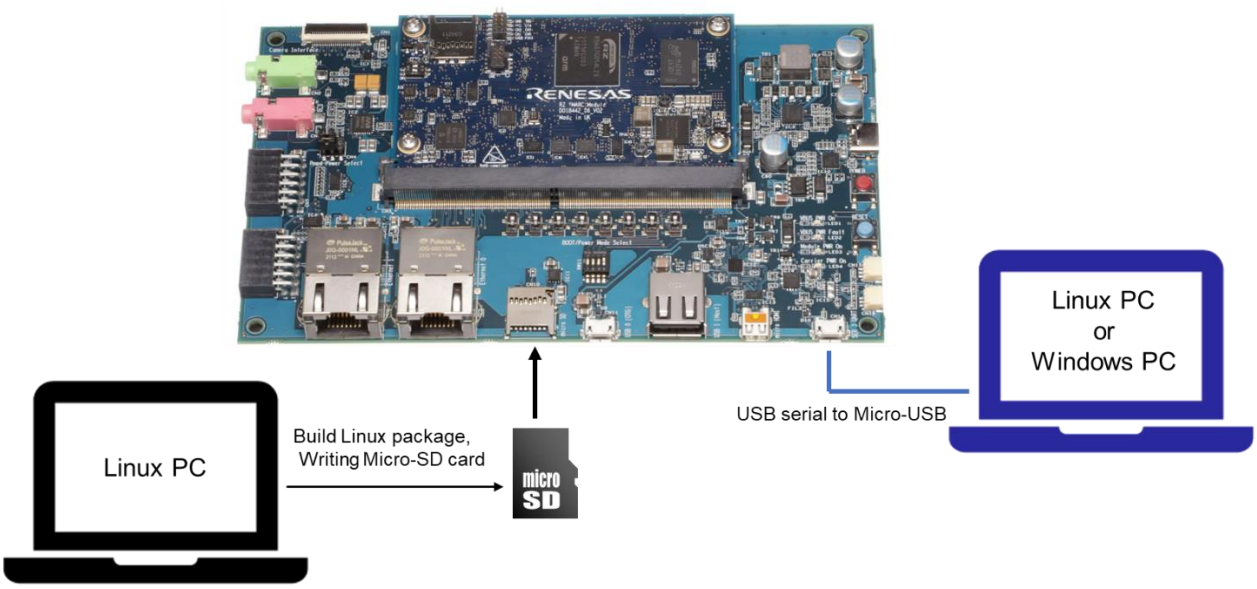


Figure 1. Recommend environment

Table 2. Equipment and Software for Developing Environments of RZ MPU Linux Platform

Equipment		Description
Linux Host PC		Used as build/debug environment 200GB free space on HDD or SSD is necessary
	OS	Ubuntu 22.04 LTS 64 bit OS must be used.
Windows Host PC		Used as debug environment, controlling with terminal software
	OS	Windows 11
	Terminal software	Used for controling serial console of the target board Tera Term (latest version) is recommended Available at Releases TeraTermProject/teraterm(github.com)
	VCP Driver	Virtual COM Port driver which enables to communicate Windows Host PC and the target board via USB which is virtually used as serial port. Available at: http://www.ftdichip.com/Drivers/VCP.htm
USB serial to micro–USB Cable		Serial communication (UART) between the Evaluation Board Kit and Windows PC. The type of USB serial connector on the Evaluation Board Kit is Micro USB type B.
micro–SD Card		Use to boot the system, and store applications.

Most bootable images VLP supports can be built on an “offline” environment.
The word “offline” means an isolated environment which does not connect to any network. Since VLP includes all necessary source codes of OSS except for the Linux kernel, VLP can always build images in this “offline” environment without affected from changes of repositories of OSS. Also, this “offline” environment reproduces the same images as the images which were verified by Renesas.

Below images can be built “offline”.

- core-image-minimal
- core-image-weston (including the SDK build)

2. Build Instructions

2.1 Required Host OS

⚠ The VLP is only built in **Ubuntu 22.04**

Ubuntu 22.04 is required to build the VLP. This is because it was the only host operating system tested and is a specific requirement for Yocto 5.0 (scarthgap).

2.2 Building images

This section describes the instructions to build the VLP.
Before starting the build, run the command below on the Linux Host PC to install packages used for building the VLP.

```
$ sudo apt-get update
$ sudo apt install build-essential chrpath cpio debianutils diffstat file gawk \
gcc git iputils-ping libacl1 liblz4-tool locales python3 python3-git \
python3-jinja2 python3-pexpect python3-pip python3-subunit socat texinfo unzip \
wget xz-utils zstd
```

Please refer to the URL below for detailed information:

- https://docs.yoctoproject.org/5.0.9/brief-yoctoprojectqs/

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 patches.**

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

Copy all files obtained from Renesas into your Linux Host PC prior to the steps below. The directory which you put the files in is described as <package download directory> in the build instructions.

(1) Set the environment variable of the package

Set the version number of the package you are using as an environment variable. The following is an example, so please rewrite it to match your package.

```
$ PACKAGE_VERSION=4.0.1
```

Note) When you download the update packages, for example, Verified Linux Package v4.0.0-update1, please remove '-update1' and set PACKAGE_VERSION as '4.0.0'.

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

Run the commands below. The name and the place of the working directory can be changed as necessary.

```
$ mkdir ~/rz_vlp_v${PACKAGE_VERSION}
$ cd ~/rz_vlp_v${PACKAGE_VERSION}
$ cp ../<package download directory>/*.zip .
$ unzip ./RTK0EF0045Z0035AZJ-v${PACKAGE_VERSION}.zip
$ tar zxvf ./RTK0EF0045Z0035AZJ-v${PACKAGE_VERSION}/rz_vlp_v${PACKAGE_VERSION}.\
tar.gz
```

Note) Please note that your building environment must have 200GB of free hard drive space to complete the minimum build. The Yocto VLP 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.

Note) If you have a board with the early silicon version, please refer to the **2.3** Notes and apply the patch files during this step. Please also confirm how to check which version you use.

(3) Enable Graphics and Video Codec

The graphics package and the video codec package can be used at the same time. And one of the packages can be used.

Please check the website page below for available combination with Yocto recipe package.

RZ MPU Verified Linux Package [6.1-CIP] | Renesas

The following operations are the EN packages. If you use the JP package, replace EN with JP.

For RZ/G2L and RZ/G2LC

The graphics package and the video codec package can be used at the same time. And one of the packages can be used.

If you want to enable the Graphics on RZ/G2L and RZ/G2LC when building **core-image-weston**, please copy the Graphics package (RTK0EF0045Z14001ZJ-<version>_EN.zip or RTK0EF0045Z14001ZJ-<version>_JP.zip) to working directory and run the commands below. If you build core-image-minimal, please ignore this step.

```
$ unzip ./RTK0EF0045Z14001ZJ-<version>_EN.zip
$ tar zxvf ./RTK0EF0045Z14001ZJ-<version>_EN/meta-rz-features_graphics_<version>.\
tar.gz
```

For RZ/G2L

If you want to enable the video codec on RZ/G2L when building **core-image-weston**, please copy the video codec package (RTK0EF0045Z16001ZJ-<version>_EN.zip or RTK0EF0045Z16001ZJ-v<version>_JP.zip) to working directory and run the commands below.

```
$ unzip ./RTK0EF0045Z16001ZJ-<version>_EN.zip
$ tar zxvf ./RTK0EF0045Z16001ZJ-<version>_EN/meta-rz-features_codec_<version>.tar.gz
```

(4) 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/rz-conf/ source \
poky/oe-init-build-env build
```

(5) Add layers

Please follow the steps below to add the layers you need. The steps add the settings to bblayers.conf.

- **Graphics**: Please run the command below if you need the Graphics library.

```
$ bitbake-layers add-layer ../meta-rz-features/meta-rz-graphics
```

- **Video Codec**: Please run the command below if you need the video codec library.

```
$ bitbake-layers add-layer ../meta-rz-features/meta-rz-codecs
```

- **Docker**: Please run the commands below if you want to include Docker. This means running Docker on the RZ board, not as using Docker as part of your build environment.

```
$ bitbake-layers add-layer ../meta-openembedded/meta-networking
$ bitbake-layers add-layer ../meta-openembedded/meta-filesystems
$ bitbake-layers add-layer ../meta-virtualization
```

(6) Decompress OSS files to “build” directory (Optional)

Run the commands below. This step is optional, and you can proceed to step (7) if an "offline" environment isn't necessary. This '7z' command will decompress all the OSS packages. For example, this step uses the path "~/rz_vlp_v${PACKAGE_VERSION}/build/" to decompress the open source software packages. You can choose any path for decompression.

```
$ cp ../../<package download directory>/*.7z .
$ 7z x oss_pkg_rz_v${PACKAGE_VERSION}.7z
```

Note) If you skip this step, the bitbake command will download all source codes from the repositories of each OSS over the internet. Please be aware that if you are not in an "offline" environment, the building might fail due to unexpected changes in the OSS repositories.
Open source software packages include all the source codes of the OSS components. These are the exact versions of the OSS used during the verification of VLP. If you are simply evaluating VLP and the RZ/G2L group, using these open source software packages are not necessary. Generally, all the software can be built without these files if your building machine has an internet connection.
Open source software packages are necessary for an "offline" environment. An "offline" environment is defined as an isolated environment without any network connection. VLP can consistently build images in this "offline" environment using these packages, unaffected by any modifications in the original OSS repositories. Furthermore, this "offline" environment always produces the same images that Renesas verified. Keep in mind that building without these open-source software packages could lead to the use of different source codes than those used by Renesas, due to potential unexpected changes in the OSS repositories.

After the above procedure is completed, the “offline” environment is ready. If you want to prevent network access, please change the line in the “~/rz_vlp_v${PACKAGE_VERSION}/build/conf/local.conf” as below:

```
BB_GENERATE_MIRROR_TARBALLS = "1"
BB_GENERATE_SHALLOW_TARBALLS = "1"
BB_GIT_SHALLOW = "1"
BB_GIT_SHALLOW_DEPTH = "1"

BB_NO_NETWORK = "1"

INHERIT += "own-mirrors"
SOURCE_MIRROR_URL = "file://<build directory>/own-mirror"
```

<build directory> should be replaced with the full path to ”~/rz_vlp_v${PACKAGE_VERSION}/build”.

Example:

```
SOURCE_MIRROR_URL = "file:///home/renesas/rz_vlp_v4.0.1/build/own-mirror"
```

Then other build PC also can refer the same “<build directory>/own-mirror” instead decompress the OSS file by themselves.

(7) 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=<board> bitbake <image name>
```

<board> can be selected by referring to Table 3.

Table 3. List of platforms and the boards

Renesas MPU	<board>
RZ/G2L	smarc-rzg2l
RZ/G2LC	smarc-rzg2lc
RZ/G2UL	smarc-rzg2ul

<image name> can be selected below. Please refer to Table 4 for supported image details.

- core-image-minimal
- core-image-weston

Table 4. Supported images of VLP

Image name	Target devices	Purpose
core-image-minimal	RZ/G2L, RZ/G2LC, RZ/G2UL	Minimal set of components
core-image-weston	RZ/G2L, RZ/G2LC, RZ/G2UL	Standard image with graphics support (*)

(*) RZ/G2UL can build core-image-weston but does not support codec and graphic packages

After the building 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 listed in Table 5 will be generated by the bitbake command and will be in the **build/tmp/deploy/images** directory.

Table 5. Image files for RZ/G2L, RZ/G2LC and RZ/G2UL

RZ/G2L	**Linux kernel**	Image-smarc-rzg2l.bin
	Device tree file	r9a07g044l2-smarc.dtb
	root filesystem	*<image name>*-smarc-rzg2l.rootfs.tar.bz2
	Bootloaders	・bl2_bp_esd-smarc-rzg2l_pmic.bin ・bl2_bp_mmc-smarc-rzg2l_pmic.srec ・bl2_bp_spi-smarc-rzg2l_pmic.srec ・fip-smarc-rzg2l_pmic.srec
	Flash Writer	Flash_Writer_SCIF_RZG2L_SMARC_PMIC_DDR4_2GB_1PCS.mot
	SD image (wic)	*<image name>*-smarc-rzg2l.rootfs.wic.gz *<image name>*-smarc-rzg2l.rootfs.wic.bmap
	Software Bill of Materials (SBOM)	*<image name>*-smarc-rzg2l.rootfs.spdx.tar.zst
RZ/G2LC	**Linux kernel**	Image-smarc-rzg2lc.bin
	Device tree file	r9a07g044c2-smarc.dtb
	root filesystem	*<image name>*-smarc-rzg2lc.rootfs.tar.bz2
	Bootloaders	・bl2_bp_esd-smarc-rzg2lc.bin ・bl2_bp_mmc-smarc-rzg2lc.srec ・bl2_bp_spi-smarc-rzg2lc.srec ・fip-smarc-rzg2lc.srec
	Flash Writer	・Flash_Writer_SCIF_RZG2LC_SMARC_DDR4_1GB_1PCS.mot
	SD image (wic)	*<image name>*-smarc-rzg2lc.rootfs.wic.gz *<image name>*-smarc-rzg2lc.rootfs.wic.bmap
	Software Bill of Materials (SBOM)	*<image name>*-smarc-rzg2lc.rootfs.spdx.tar.zst
RZ/G2UL	**Linux kernel**	Image-smarc-rzg2ul.bin
	Device tree file	r9a07g043u11-smarc.dtb
	root filesystem	*<image name>*-smarc-rzg2ul.rootfs.tar.bz2
	Bootloaders	・bl2_bp_esd-smarc-rzg2ul.bin ・bl2_bp_mmc-smarc-rzg2ul.srec ・bl2_bp_spi-smarc-rzg2ul.srec ・fip-smarc-rzg2ul.srec
	Flash Writer	・Flash_Writer_SCIF_RZG2UL_SMARC_DDR4_1GB_1PCS.mot
	SD image (wic)	*<image name>*-smarc-rzg2ul.rootfs.wic.gz *<image name>*-smarc-rzg2ul.rootfs.wic.bmap
	Software Bill of Materials (SBOM)	*<image name>*-smarc-rzg2ul.rootfs.spdx.tar.zst

⚠ To boot the EVK with the new VLP version, rewrite the Bootloaders.

2.3 Notes

(1) Early version device

When you use the **early** version of the RZ/G2L LSI, please run the commands below to apply the patch files after step (1) in the section 2.2.

```
$ cd ~/rz_vlp_v${PACKAGE_VERSION}/meta-renesas
$ patch -p1 < ../extra/0001-trusted-firmware-a-add-rd-wr-64-bit-reg-workaround.patch
$ patch -p1 < ../extra/0002-rz-common-linux-renesas-add-WA-GIC-access-64bit.patch
```

Note) If you want to know which version of the RZ/G2L LSI you use, please check the LSI on the board. When "2050KC002" is printed on the LSI, you use the early version.

(2) Exclude GPLv3 packages

Run the commands below to download meta-rz-nogplv3.

```
$ cd ~/rz_vlp_v${PACKAGE_VERSION}/
$ git clone https://github.com/renesas-rz/meta-rz-nogplv3.git
$ cd meta-rz-nogplv3/
$ git checkout 7c9d35357d95a9603da412aa5a8ecc3c8f723658
$ cd ..
```

Then run the below command to add the layer.

```
$ cd build
$ bitbake-layers add-layer ../meta-rz-nogplv3
```

If you do not run below command, please run below command first.

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

Note) Renesas provides no guarantees or support whatsoever. The customer is solely responsible for cloning the source from GitHub at their own discretion.

(3) Real time performance

If you want to use the kernel which improves the real-time performance, please add the line below to the file " ~/rz_vlp_v${PACKAGE_VERSION}/build/conf/local.conf".

```
PREFERRED_PROVIDER_virtual/kernel = "linux-renesas-rt"
```

(4) Locale Issue

If bitbake shows below error log after starting building.

```
Please make sure locale 'en_US.UTF-8' is available on your system
```

Please run below command on the build PC.

```
$ sudo apt-get install locales
:
:
Generation complete.

$ sudo dpkg-reconfigure locales
:
:
```

```
<Please choose case "en_US.UTF-8 UTF-8", then default the language to en_US.UTF-8>
:
This will select the default language for the entire system. If this system is a mul
ti-user system where not all users are able to speak the default language, they will
 experience difficulties.


  1. None  2. C.UTF-8  3. en_US.UTF-8
Default locale for the system environment: 3


Generating locales (this might take a while)...
  en_US.UTF-8... done
Generation complete.
$ sudo locale-gen en_US.UTF-8
Generating locales (this might take a while)...
  en_US.UTF-8... done
Generation complete.
$ sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8
$ export LANG=en_US.UTF-8
```

3. Preparing the SD Card

You can prepare the microSD card by the following method.

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
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xxxxxxxxxx
$ umount /dev/sdb1
$ umount /dev/sdb2
```

Expand disk image.

```
$ sudo bmaptool copy <wic image>.wic.gz /dev/sdb
```

The file nemes of *<wic image>* is listed in the **Table 5**.

Table 6. File and directory in the micro SD card

Type/Number	Filesystem	Contents
Primary #1	FAT32	Flash_Writer_SCIF_<device>.mot bl2_bp_mmc-smarc-<device>.srec bl2_bp_spi-smarc-<device>.srec bl2_bp_esd-smarc-<device>.bin fip-smarc-<device>.srec fip-smarc-<device>.bin
Primary #2	Ext4	./ ├── bin ├── boot │ ├── Image │ └── <device>-smarc.dtb ├── dev ├── etc ├── home ├── lib ├── media ├── mnt ├── proc ├── run ├── sbin ├── srv ├── sys ├── tmp ├── usr └── var

4. Reference Board Setting

4.1 Preparation of Hardware and Software

The following environment of Hardware and Software is used in the evaluation.

Hardware preparation (Users should purchase the following equipment.):

- USB Type-C cable compatible with USB PD. (e.g. AK-A8485011 (manufactured by Anker))
- USB PD Charger 15W (5V 3.0A) or more. (e.g. PowerPort III 65W Pod (manufactured by Anker))
- USB Type-microAB cable (Any cables)
- micro HDMI cable (Any cables)
- PC Installed FTDI VCP driver and Terminal software (Tera Term) (*1)

(*1) Please install the FTDI driver that can be following website (https://www.ftdichip.com/Drivers/VCP.htm).

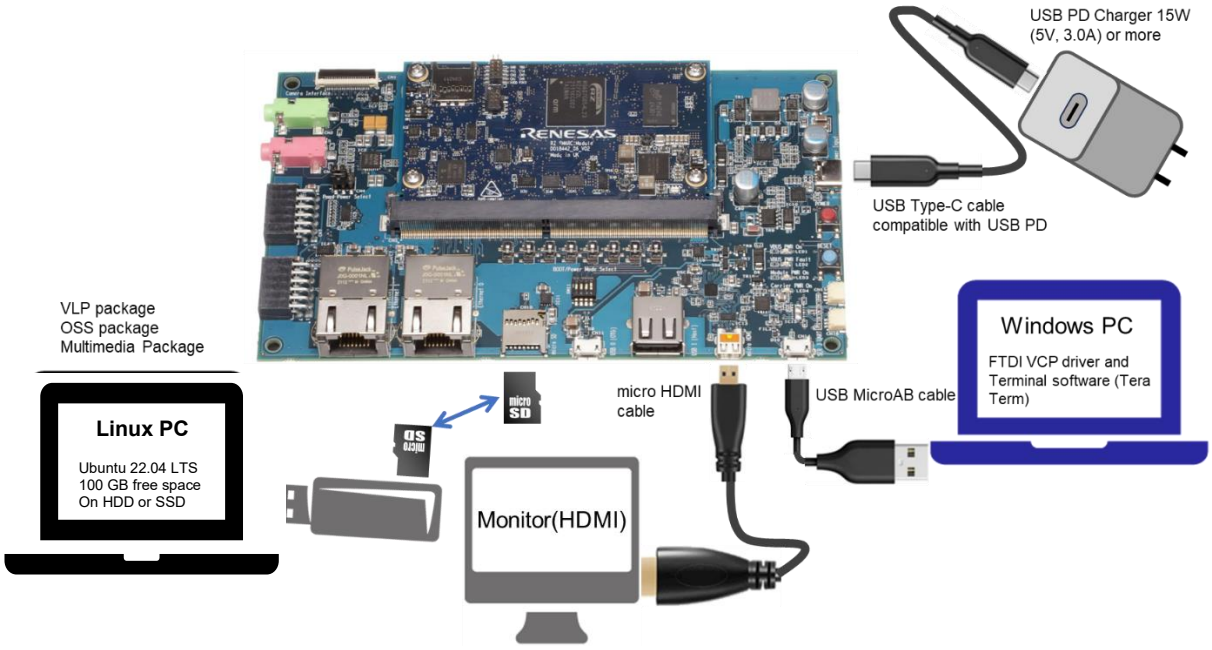


Figure 2. Operating environment

4.1.1 How to set boot mode and input voltage

Please set the SW11 settings as follows.

- Pin no1 to no3 of the SW11 is used to control boot mode of RZ/G2L, RZ/G2LC and RZ/G2UL.
- Pin no4 of the SW11 is used to control the input voltage from power charger to 5V or 9V. Please use a 5V setting as the initial setting.

SW11-1	OFF
SW11-2	ON
SW11-3	OFF
SW11-4	ON

Please select boot mode as below figures.

SCIF Download Mode(*1)

SW11-1	OFF
SW11-2	ON
SW11-3	OFF
SW11-4	ON

QSPI Boot (1.8V) Mode

SW11-1	OFF
SW11-2	OFF
SW11-3	OFF
SW11-4	ON

eMMC Boot (1.8V) Mode

SW11-1	ON
SW11-2	OFF
SW11-3	OFF
SW11-4	ON

eSD Boot Mode

SW11-1	ON
SW11-2	ON
SW11-3	OFF
SW11-4	ON

Please select input voltage setting as below.

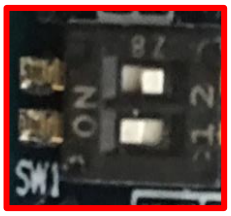
SW11-4	Input voltage selection
OFF	Input 9V
ON	Input 5V

4.1.2 How to set SW1

Please set the SW1 settings to follows for RZ/G2L Smarc Module Board.

For RZ/G2LC and RZ/G2UL, please refer to User's Manual: Hardware (RZ/G2LC-EVKIT - Evaluation Board Kit for RZ/G2LC MPU | Renesas, RZ/G2UL-EVKIT - Evaluation Board Kit for RZ/G2UL MPU | Renesas).

- Pin no. 1 of the SW1 is used to select the JTAG debug mode or not.
  JTAG is not used, so set SW1-1 to normal operation mode.
- Pin no2 of the SW1 is used to select the eMMC or microSD mode. Please set SW1-2 to eMMC mode.



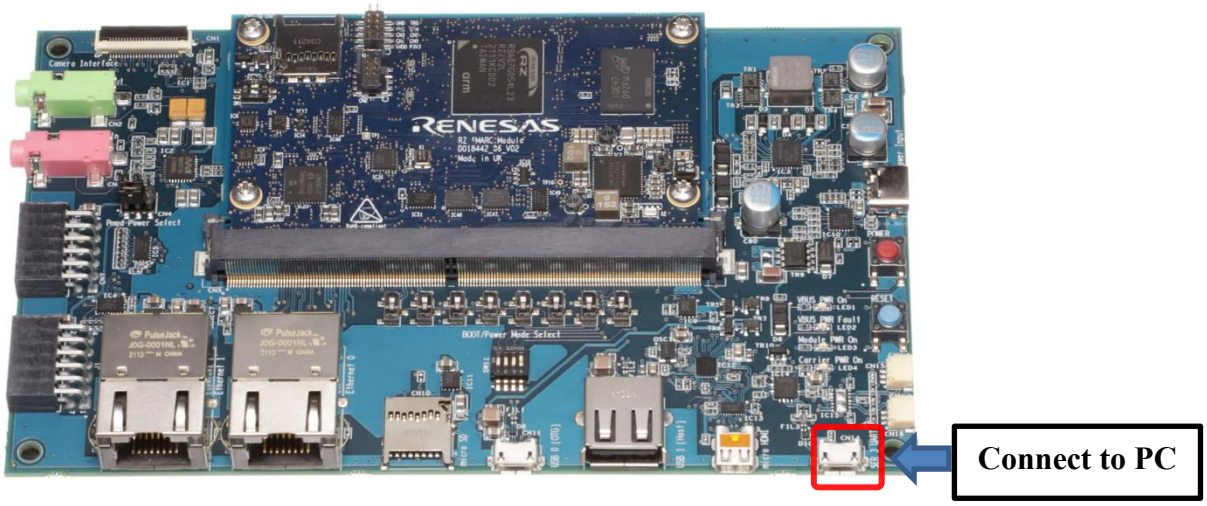
SW1-2	OFF
SW1-1	ON

No.	Description	Setting	Function
SW1-2	Selection SD/MMC	OFF	Select the eMMC memory
		ON	Select the SD card
SW1-1	DEBUGEN	OFF	JTAG debugging
		ON	Normal operation

The selection of microSD slot and eMMC on the SMARC module is exclusive

4.1.3 How to use debug serial (console ouput)

Please connect USB Type-micro-B cable to CN14.



CN14:USB Type-micro-B Connector

Figure 3. Connecting console for debug

4.2 Startup Procedure

This section describes how to write Flash writer and bootloaders using Windows PC. For descriptions how to write using Linux PC, please refer to 8.8.

4.2.1 Power supply

1. Connect USB-PD Power Charger to USB Type-C Connector (CN6).
2. LED1(VBUS Power ON) and LED3 (Module PWR On) light up.

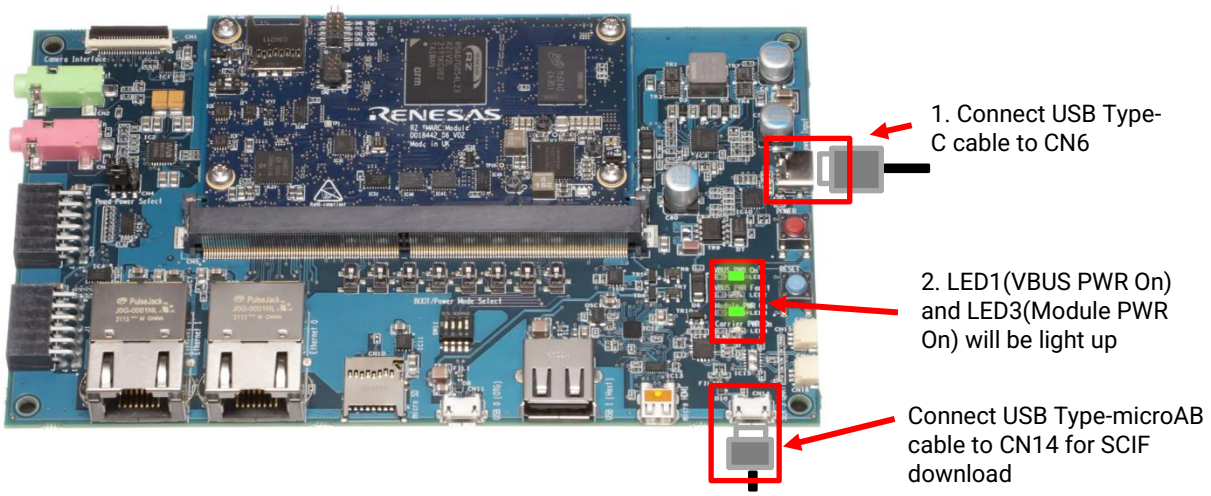


Figure 4. Connecting Power Supply

3. Press the power button (SW9) to turn on the power.
   Note: When turning on the power, press and hold the power button for 1 second.
   When turn off the power, press and hold the power button for 2 seconds
4. LED4(Carrier PWR On) lights up.

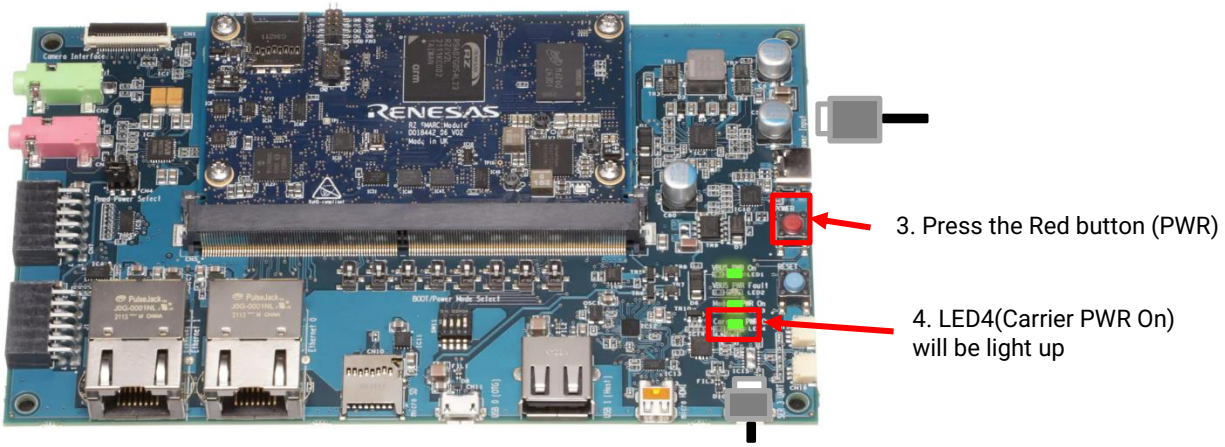


Figure 5. Power ON

4.2.2 Building files to write

The evaluation boards use the files in the Table 7 as boot loaders. The boot loaders files are generated by the build instruction in section 2. Please refer to Table 5. Once built, copy these files to a PC which runs serial terminal software.

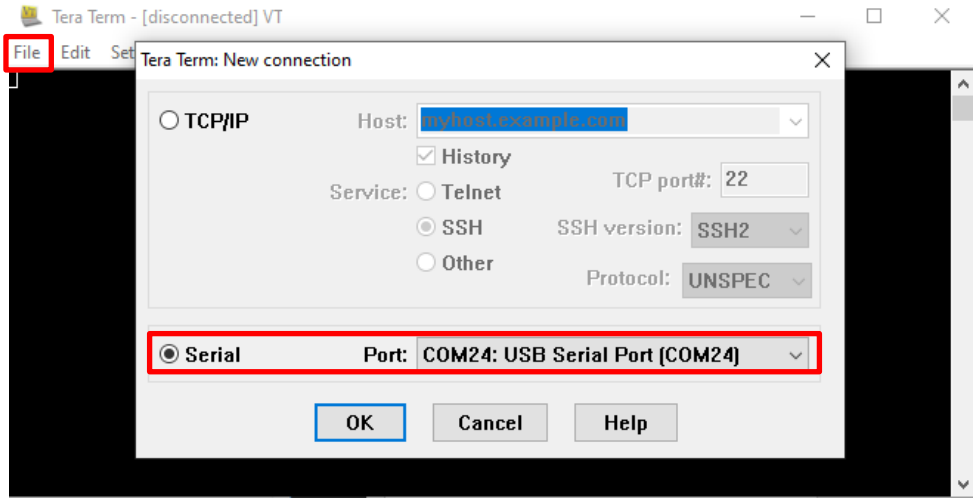
Table 7. File names of Boot loader

Board	File name of Boot loader
RZ/G2L Evaluation Board Kit	• bl2_bp_spi-smarc-rzg2l_pmic.srec • fip-smarc-rzg2l_pmic.srec
RZ/G2LC Evaluation Board Kit	• bl2_bp_spi-smarc-rzg2lc.srec • fip-smarc-rzg2lc.srec
RZ/G2UL Evaluation Board Kit	• bl2_bp_spi-smarc-rzg2ul.srec • fip-smarc-rzg2ul.srec

4.2.3 Settings

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

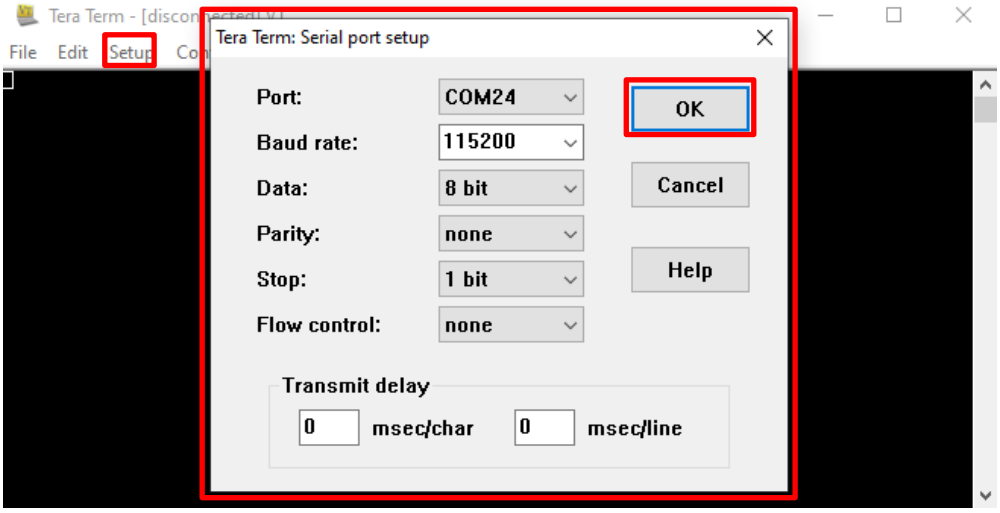
1. Bring up the terminal software and select the “File” > “New Connection” to set the connection on the software.



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

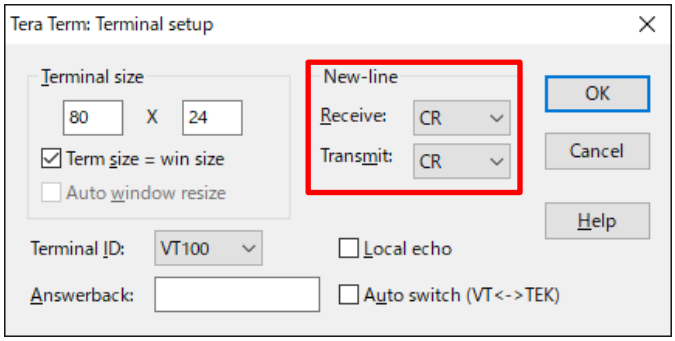
Set the settings about serial communication protocol on a terminal software as below:

- Speed: 115200 bps
- Data: 8bit
- Parity: None
- Stop bit: 1bit
- Flow control: None



Select the “Setup” > “Terminal” to set the new-line code.

- New-line:”CR” or “AUTO”



3. To set the board to SCIF Download mode, set the SW11 as below (please refer 2.1.2):

Table 8. SW11

1	2	3	4
OFF	ON	OFF	ON

4. After finishing all settings, when pressed the reset button SW10, the messages below are displayed on the terminal.

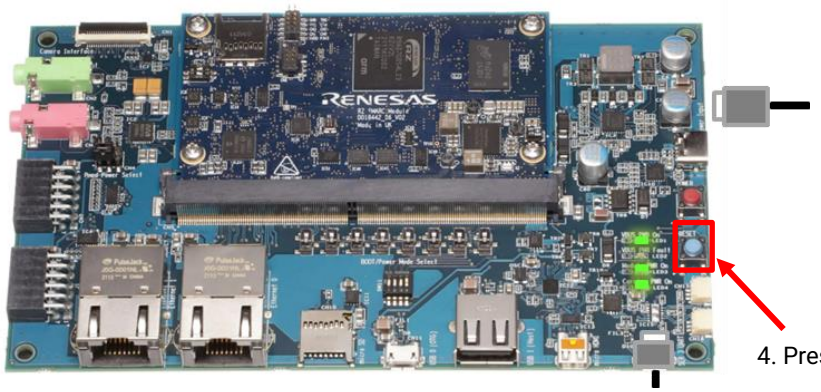
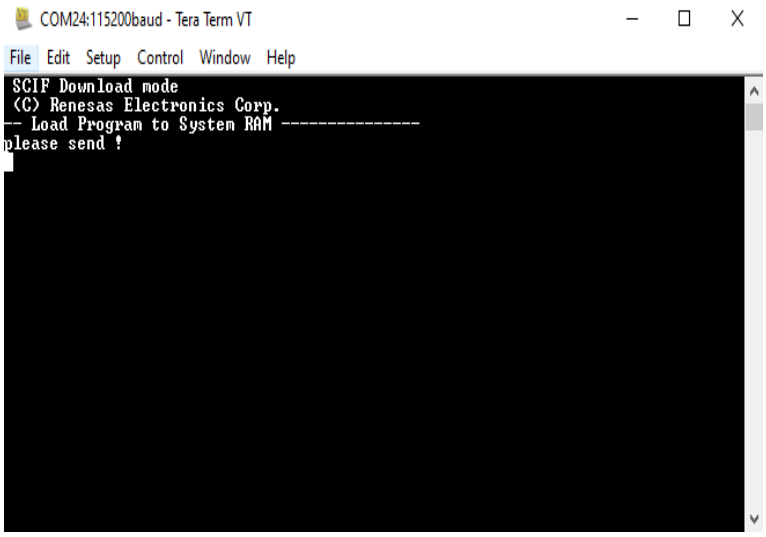


Figure 6. Press the RESET button



4.3 Download Flash Writer to RAM

Copy the Flash Writer and Bootloader files from the built-generated Image directory or from the microSD card where the wic image was deployed, as shown in **Table 5**, to your PC.

Turn on the power of the board by pressing SW9. The messages below are shown on the terminal.

```
 SCIF Download mode
 (C) Renesas Electronics Corp.

-- Load Program to SystemRAM ---------------
please send !
```

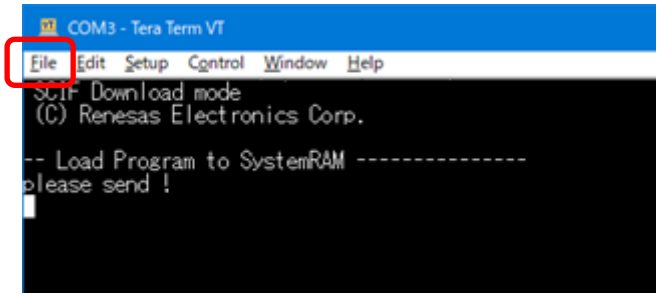
Send an image of Flash Writer using terminal software after the message “please send !” is shown. Please refer to the Table 9 below to know which file name of Flash Writer should be sent.

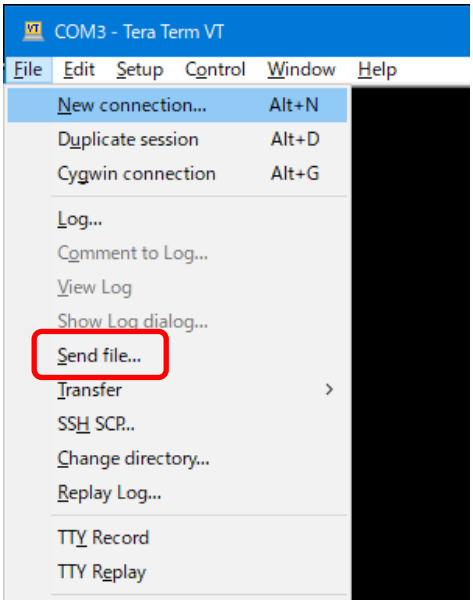
Table 9. File names of Flash Writer

Board name	File name of Flash Writer
RZ/G2L Evaluation Board Kit	Flash_Writer_SCIF_RZG2L_SMARC_PMIC_DDR4_2GB_1PCS.mot
RZ/G2LC Evaluation Board Kit	Flash_Writer_SCIF_RZG2LC_SMARC_DDR4_1GB_1PCS.mot
RZ/G2UL Evaluation Board Kit	Flash_Writer_SCIF_RZG2UL_SMARC_DDR4_1GB_1PCS.mot

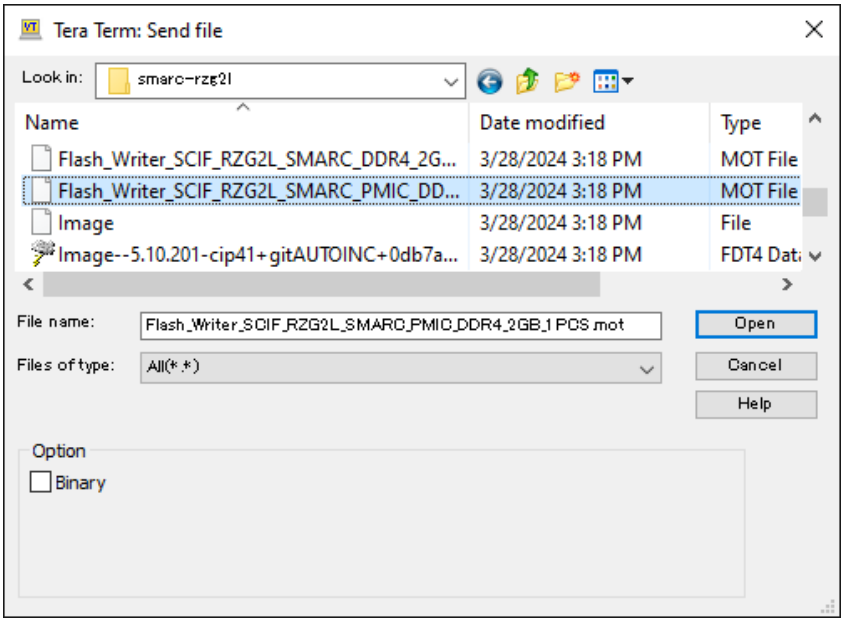
Below is a sample procedure with Tera Term.

Open a “Send file” dialog by selecting “File” → “Sendfile” menu.

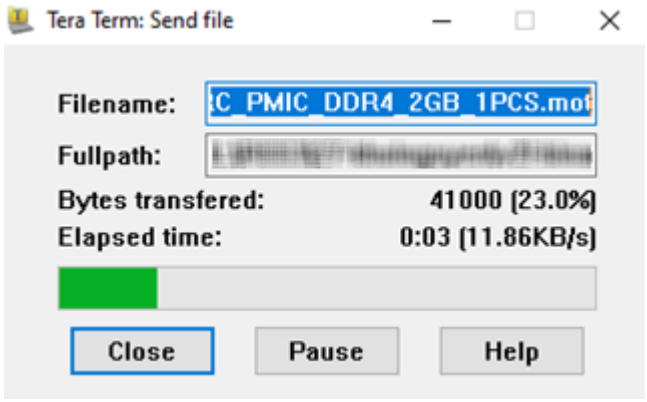




Then, select the image to be send and click “Open” button.



The image will be sent to the board via serial connection.



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

```
Flash writer for RZ/G2 Series V1.02 Nov.15,2021
 Product Code : RZ/G2L
>
```

4.4 Write the Bootloader

For the boot operation, two boot loader files need to be written to the target board. Corresponding bootloader files and specified address information depend on each target board as described in Table 10.

Before writing the loader files, change the Flash Writer transfer rate from default (115200bps) to high speed (921600bps) with “SUP” command of Flash Writer.

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

After “SUP” command, change the serial communication protocol speed from 115200bps to 921600bps as well by following the steps described in 4.2.3, and push the enter key.

Next, use “XLS2” command of Flash Writer to write boot loader binary files. This command receives binary data from the serial port and writes the data to a specified address of the Flash ROM with information where the data should be loaded on the address of the main memory.

For example, this part describes how to write boot loader files in the case of RZ/G2L Evaluation Board Kit:

```
>XLS2
===== Qspi writing of RZ/G2 Board Command =============
Load Program to Spiflash
Writes to any of SPI address.
 Micron : MT25QU512
Program Top Address & Qspi Save Address
===== Please Input Program Top Address ============
  Please Input : H'11E00

===== Please Input Qspi Save Address ===
  Please Input : H'00000
Work RAM(H'50000000-H'53FFFFFF) Clear....
please send ! ('.' & CR stop load)
```

Send the data of “bl2_bp_spi-smarc-rzg2l_pmic.srec” from terminal software after the message “please send !” is shown.

After successfully downloading the binary, messages like below are shown on the terminal.

```
SPI Data Clear(H'FF) Check :H'00000000-0000FFFF Erasing..Erase Completed
SAVE SPI-FLASH.......
======= Qspi  Save Information  =================
 SpiFlashMemory Stat Address : H'00000000
 SpiFlashMemory End Address  : H'00009A80
============================================================
```

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

In case a message to prompt to clear data like above, please enter “y”.

Next, write another loader file by using XLS2 command again.

```
>XLS2
===== Qspi writing of RZ/G2 Board Command =============
Load Program to Spiflash
Writes to any of SPI address.
 Micron : MT25QU512
```

```
Program Top Address & Qspi Save Address
===== Please Input Program Top Address ============
  Please Input : H'00000

===== Please Input Qspi Save Address ===
  Please Input : H'20000
Work RAM(H'50000000-H'53FFFFFF) Clear....
please send ! ('.' & CR stop load)
```

Send the data of “fip-smarc-rzg2l_pmic.srec” from terminal software after the message “please send !” is shown.

After successfully downloading the binary, messages like below are shown on the terminal.

```
SPI Data Clear(H'FF) Check :H'00000000-0000FFFF Erasing..Erase Completed
SAVE SPI-FLASH.......
======= Qspi  Save Information  =================
 SpiFlashMemory Stat Address : H'00020000
 SpiFlashMemory End Address  : H'000F64CF
============================================================
```

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

In case a message to prompt to clear data like above, please enter “y”.

After writing two loader files normally, change the serial communication protocol speed from 921600 bps to 115200 bps by following the steps described in 4.2.3.

Finally, turn off the power of the board by pressing SW9.

Table 10. Address for sending each loader binary file

RZ/G2L Evaluation Board Kit

File name	Address to load to RAM	Address to save to ROM
bl2_bp_spi-smarc-rzg2l_pmic.srec	11E00	00000
fip-smarc-rzg2l_pmic.srec	00000	20000

RZ/G2LC Evaluation Board Kit

File name	Address to load to RAM	Address to save to ROM
bl2_bp_spi-smarc-rzg2lc.srec	11E00	00000
fip-smarc-rzg2lc.srec	00000	20000

RZ/G2UL Evaluation Board Kit

File name	Address to load to RAM	Address to save to ROM
bl2_bp_spi-smarc-rzg2ul.srec	11E00	00000
fip-smarc-rzg2ul.srec	00000	20000

From VLP v4.0.1, QSPI Save address for fip-xxx.srec is changed to H’20000 with the latest update from TF-A.

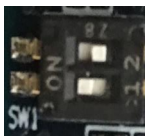
4.5 Change Back to Normal Boot Mode

To set the board to SPI Boot mode, set the SW11 as below:

Table 11. SW11

1	2	3	4
OFF	OFF	OFF	ON

Set the SW1 on SoM module to eMMC mode. Please refer to the section 4.1.2 for RZ/G2L.



For RZ/G2LC and RZ/G2UL EVK, please refer to User's Manual: Hardware (RZ/G2LC-EVKIT - Evaluation Board Kit for RZ/G2LC MPU | Renesas, RZ/G2UL-EVKIT - Evaluation Board Kit for RZ/G2UL MPU | Renesas).

Turn on the board's power by pressing the power red button SW9. Press the blue button SW10 to reset.

Stop the autoboot procedure by pressing the enter key on your keyboard before the countdown stops.

If you are not fast enough, you can try again by pressing the blue reset button.

```
NOTICE:  BL2: v2.10.5(release):2.10.5/rz_1.1.0-dirty
NOTICE:  BL2: Built : 13:04:30, Sep 25 2025
NOTICE:  BL2: RZ/G2L
NOTICE:  BL2: SYS_LSI_MODE: 0x3
NOTICE:  BL2: SYS_LSI_DEVID: 0x4841c447
NOTICE:  BL2: SYS_LSI_PRR: 0x0
NOTICE:  BL2: Booting BL31
NOTICE:  BL31: v2.10.5(release):2.10.5/rz_1.1.0-dirty
NOTICE:  BL31: Built : 13:04:30, Sep 25 2025


U-Boot 2024.07 (Sep 22 2025 - 10:51:32 +0000)

CPU:   Renesas Electronics CPU rev 1.0
Model: smarc-rzg2l
DRAM:  1.9 GiB
SF: Detected mt25qu512a with page size 256 Bytes, erase size 4 KiB, total 64 MiB
Core:  33 devices, 19 uclasses, devicetree: fit
WDT:   watchdog@0000000012800800
WDT:   Started watchdog@12800800 with servicing every 1000ms (60s timeout)
MMC:   sd@11c00000: 0, sd@11c10000: 1
Loading Environment from MMC... Reading from MMC(0)... OK
In:    serial@1004b800
Out:   serial@1004b800
Err:   serial@1004b800
WDT:   Started watchdog@12800800 with servicing every 1000ms (60s timeout)
U-boot WDT started!
```

```
Net:   eth0: ethernet@11c20000
Warning: ethernet@11c30000 (eth1) using random MAC address - fa:72:9e:48:2b:2d
, eth1: ethernet@11c30000
Hit any key to stop autoboot:  0
=>
```

Following the messages above, many warning messages will be shown. These warnings are eliminated by setting correct environment variables. Please set default 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
=>
```

5. Booting and Running Linux

Set microSD card to slot on carry board.

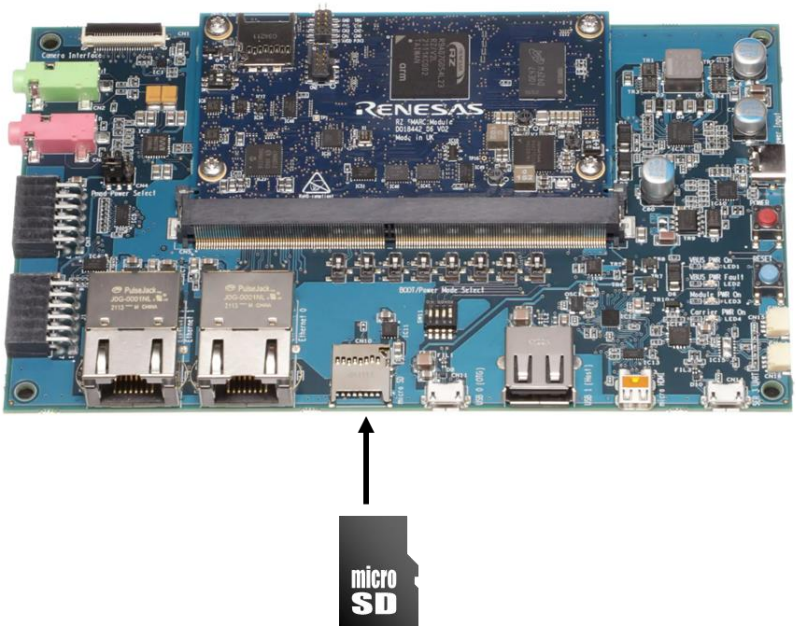


Figure 7. Set micro SD card to SMARC-EVK

Now the board can boot up normally. Please turn off and on the power again to boot up the board.

5.1 Power on the board and Startup Linux

After obtaining your reference board, please be sure to follow the document and write the bootloaders to the Flash ROM before starting the evaluation.

Before booting the board, please be sure to confirm the bootloaders which are built with your VLP are written to your board.

Turn the power off and on by pressing red button SW9 again to boot up the board. When “log in” displayed, enter “root” to login. (A password is not required).

```
U-Boot 2024.07 (Sep 22 2025 - 10:51:32 +0000)

CPU:   Renesas Electronics CPU rev 1.0
Model: smarc-rzg2l
DRAM:  1.9 GiB
SF: Detected mt25qu512a with page size 256 Bytes, erase size 4 KiB, total 64 MiB
Core:  33 devices, 19 uclasses, devicetree: fit
WDT:   watchdog@0000000012800800
WDT:   Started watchdog@12800800 with servicing every 1000ms (60s timeout)
MMC:   sd@11c00000: 0, sd@11c10000: 1
Loading Environment from MMC... Reading from MMC(0)... OK
In:    serial@1004b800
Out:   serial@1004b800
Err:   serial@1004b800
WDT:   Started watchdog@12800800 with servicing every 1000ms (60s timeout)
U-boot WDT started!
Net:   eth0: ethernet@11c20000
Warning: ethernet@11c30000 (eth1) using random MAC address - fa:72:9e:48:2b:2d
, eth1: ethernet@11c30000
```

```
Hit any key to stop autoboot:  0
=>
## Resetting to default environment
Card did not respond to voltage select! : -110
24373760 bytes read in 1049 ms (22.2 MiB/s)
40444 bytes read in 4 ms (9.6 MiB/s)
Moving Image from 0x48080000 to 0x48200000, end=499d0000
## Flattened Device Tree blob at 48000000
   Booting using the fdt blob at 0x48000000
   Loading Device Tree to 0000000057ff3000, end 0000000057fffdfb ... OK

Starting kernel ...

[    0.000000] Booting Linux on physical CPU 0x0000000000 [0x412fd050]
[    0.000000] Linux version 6.1.107-cip28-yocto-standard (oe-user@oe-host)
:
:
Poky (Yocto Project Reference Distro) 5.0.8 smarc-rzg2l ttySC0

smarc-rzg2l login: root
root@smarc-rzg2l:~#
```

5.2 Shutdown the Board

To power down the system, follow the step below.

Step 1. Run shutdown command

Run shutdown command on the console as below. After that, the shutdown sequence will start.

```
root@smarc-rzg2l:~# shutdown -h now
```

Note: Run this command during the power-off sequence on rootfs.

Step 2. Confirm the power-off

After executing the shutdown command, you can see " reboot: Power down" message.

Step 3. Turn off the power switch on the board

After checking the above "Carrier PWR On" LED4, turn "POWER" SW9 off.

6. Building the SDK

To build Software Development Kit (SDK), run the commands below after the steps (1) – (7) of section **2.2** are finished.

The SDK allows you to build custom applications outside of the Yocto environment, even on a completely different PC. The results of the commands below are 'installer' that you will use to install the SDK on the same PC, or a completely different PC.

```
$ cd ~/rz_vlp_v${PACKAGE_VERSION}/build
$ MACHINE=<board> bitbake core-image-weston -c populate_sdk
```

The resulting SDK installer will be in **build/tmp/deploy/sdk/**

The SDK installer will have the extension .sh
To run the installer, you would execute the following command:

```
$ sudo sh rz-vlp-glibc-x86_64-core-image-weston-cortexa55-<board>-toolchain-\
5.0.11.sh
```

<board> can be replaced as below.

RZ/G2L Evaluation Board Kit: smarc-rzg2l

RZ/G2LC Evaluation Board Kit: smarc-rzg2lc

7. Application Building and Running

This chapter explains how to make and run an application for RZ/G2L with this package.

7.1 Make an application

Here is an example of how to make an application running on VLP. The following steps will generate the “Hello World” sample application.

Note that you must build (bitbake) a core image for the target and prepare SDK before making an application. Refer to the start-up guide on how to make SDK.

7.1.1 How to extract SDK

Step 1. Install toolchain on a Host PC:

```
$ sudo sh rz-vlp-glibc-x86_64-core-image-weston-cortexa55-<board>-toolchain-\
5.0.11.sh
```

Note:

sudo is optional in case user wants to extract SDK into a restricted directory (such as: /opt/).

If the installation is successful, the following messages will appear:

```
renesas@49d1d5882435:/mnt/host$ sudo sh ./rz_vlp_v${PACKAGE_VERSION}/build/tmp/deploy/
sdk/ rz-vlp-glibc-x86_64-core-image-weston-cortexa55-<board>-toolchain-x.x.x.sh
Poky (Yocto Project Reference Distro) SDK installer version x.x.x
==========================================================================
Enter target directory for SDK (default: /opt/rz-vlp/x.x.xx):
You are about to install the SDK to "/opt/rz-vlp/x.x.xx". Proceed [Y/n]? Y
Extracting SDK.....................................................................
.....................................................done
Setting it up...done
SDK has been successfully set up and is ready to be used.
Each time you wish to use the SDK in a new shell session, you need to source the envir
onment setup script e.g.
 $ . /opt/rz-vlp/x.x.x/environment-setup-cortexa55-poky-linux
```

Step 2. Set up cross-compile environment:

```
$ source /<Location in which SDK is extracted>/environment-setup-cortexa55-poky-linux
```

Note:

User needs to run the above command once for each login session.

```
$ source /opt/poky/x.x.xx/environment-setup-cortexa55-poky-linux
```

7.1.2 How to build Linux application

Step 1. Make a directory for the application on the Linux host PC.

```
$ mkdir ~/hello_apl
$ cd ~/hello_apl
```

Step 2. Make the following three files (an application file, Makefile, and configure file) in the directory for the application.

Here, the application is made by automake and autoconf.

• main.c

```
#include <stdio.h>
/* Display "Hello World" text on terminal software */
int main(int argc, char** argv)
{
  printf("\nHello World\n");
  return 0;
}
```

• Makefile

```
APP = linux-helloworld
SRC = main.c

all: $(APP)

CC ?= gcc

# Options for development
CFLAGS = -g -O0 -Wall -DDEBUG_LOG

$(APP):
<-tab->$(CC) -o $(APP) $(SRC) $(CFLAGS)

install:
<-tab->install -D -m755 $(APP) $(DESTDIR)/home/root/$(APP)

clean:
<-tab->rm -rf $(APP)
```

Step 3. Make the application by the generated makefile.

```
$ make
```

```
$ make
aarch64-poky-linux-gcc  -mtune=cortex-a55 -fstack-protector-strong  -D_FORTIFY_SOURCE=
2 -Wformat -Wformat-security -Werror=format-security --sysroot=/opt/poky/3.1.33/sysroo
ts/aarch64-poky-linux -o linux-helloworld main.c -g -O0 -Wall -DDEBUG_LOG
```

After making, confirm that the execute application (the sample file name is “linux-helloworld”) is generated in the hello_apl folder. Also, this application must be cross-compiled for Aarch64. Mar.17.2023 4.2

Step 4. Store a sample application

The sample application could be written by the following procedure. The application should be stored in the ext4 partition.

```
$ sudo mount /dev/sdb2 /media/
$ cd /media/usr/bin
$ sudo cp ~/hello_apl/linux-helloworld .
$ sudo chmod +x linux-helloworld
```

Notes: 1. “sdb2” (above in red) may depend on using system.

2. “hello_apl” is an optional directory name to store the application.

7.2 Run a sample application

Power on the RZ/G2L Evaluation Board Kit and start the system. After booting, run the sample application with the following command.

```
smarc-rzg2l login: root
root@smarc-rzg2l:~# /usr/bin/linux-helloworld

Hello World
root@smarc-rzg2l:~#
```

Note: Refer to the start-up guide for the method of how to boot the board and system.

8. Appendix

8.1 Preparing Flash Writer

Flash Writer is built automatically when building VLP by bitbake command. Please refer to the Release Note of the RZ/G2L VLP Group to obtain a binary file of Flash Writer.

If you need the latest one, please get source code from the GitHub repository and build it according to the following instructions. In general, the new revision of reference boards requires the latest Flash Writer.

8.1.1 Preparing cross compiler

FlashWriter runs on target boards. Please get a cross compiler built by Linaro or setup Yocto SDK.

- **ARM toolchain:**

```
$ cd ~/
$ wget https://developer.arm.com/-/media/Files/downloads/gnu-a/10.2-2020.11/binrel/gcc
-arm-10.2-2020.11-x86_64-aarch64-none-elf.tar.xz
$ tar xvf gcc-arm-10.2-2020.11-x86_64-aarch64-none-elf.tar.xz
```

- **Yocto SDK:**

Build an SDK according to Release Notes and install it to a Linux Host PC. Then, enable the SDK as below.

```
$ source /opt/poky/x.x.xx/environment-setup-aarch64-poky-linux
```

8.1.2 Building Flash Writer

Get source codes of Flash Writer from the GitHub repository and check out the branch rz_g2l.

```
$ cd ~/
$ git clone https://github.com/renesas-rz/rzg2_flash_writer.git
$ cd rzg2_flash_writer
$ git checkout -b tmp 43509f2b268b0ce86288cf3c37e668d64c5d5d12
```

Build Flash Writer as an s-record file by the following commands. Please specify a target board by "BOARD" option.

- **ARM toolchain:**

```
$ export PATH=$PATH:~/gcc-arm-10.2-2020.11-x86_64-aarch64-none-elf/bin
$ export CROSS_COMPILE=aarch64-none-elf-
$ export CC=${CROSS_COMPILE}gcc
$ export AS=${CROSS_COMPILE}as
$ export LD=${CROSS_COMPILE}ld
$ export AR=${CROSS_COMPILE}ar
$ export OBJDUMP=${CROSS_COMPILE}objdump
$ export OBJCOPY=${CROSS_COMPILE}objcopy

$ make clean
$ make BOARD=<board>
```

- **Yocto SDK:**

```
$ make clean
$ make BOARD=<board>
```

Please replace *<board>* with a proper option according to the **Table 12**.

Table 12. BOARD option

Target board	BOARD option *<board>*	Image to be generated
smarc-rzg2l	RZG2L_SMARC_PMIC	Flash_Writer_SCIF_RZG2L_SMARC_PMIC_DDR4_2GB_1PCS.mot
smarc-rzg2lc	RZG2LC_SMARC	Flash_Writer_SCIF_RZG2LC_SMARC_DDR4_1GB_1PCS.mot
smarc-rzg2ul	RZG2UL_SMARC	Flash_Writer_SCIF_RZG2UL_SMARC_DDR4_1GB_1PCS.mot

8.2 How to distinguish each board

The differences between each board are as follows.

Comparison of RZ/G2L Evaluation Board Kit and RZ/V2L Evaluation Board Kit:

	RZ/G2L Evaluation Kit	RZ/V2L Evaluation Kit
IC1	RZ/G2L (R9A07G044L23GBG)	RZ/V2L (R9A07G054L23GBG)

Comparison of RZ/G2L Evaluation Board Kit and RZ/G2LC Evaluation Board Kit:

	RZ/G2L Evaluation Kit	RZ/G2LC Evaluation Kit
IC1	RZ/G2L (R9A07G044L23GBG)	RZ/G2LC (R9A07G054GBG)
IC7	MT40A1G16KD-062E:E (2GB)	MT40A512M16LY-062EIT:E (1GB)
IC23	Ethernet PHY (KSZ9131RNXC)	Not mounted
CN1	10-pin connector for ADC	Not mounted
SW1	2bit bus switch	6bit bus switch

Comparison of RZ/G2UL Evaluation Board Kit and RZ/Five Evaluation Board Kit:

	RZ/G2UL Evaluation Kit	RZ/Five Evaluation Kit
IC1	RZ/G2UL (R9A07G043U11GBG)	RZ/Five (R9A07G043F01GBG)

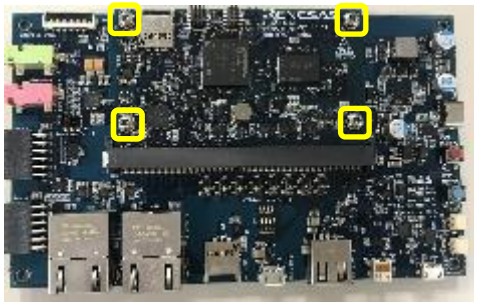
8.3 How to replace the SMARC Module Board

Please be careful when replacing the board as follows.

1. Remove the four screws.

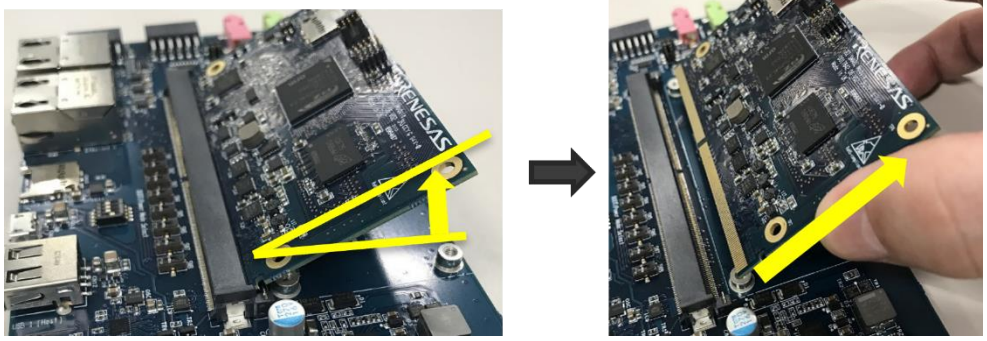
Note: The screw thread is a special shape, so be careful not to crush the screw thread.

Please recommend to prepare a torx screwdriver which is a "T6" head size.

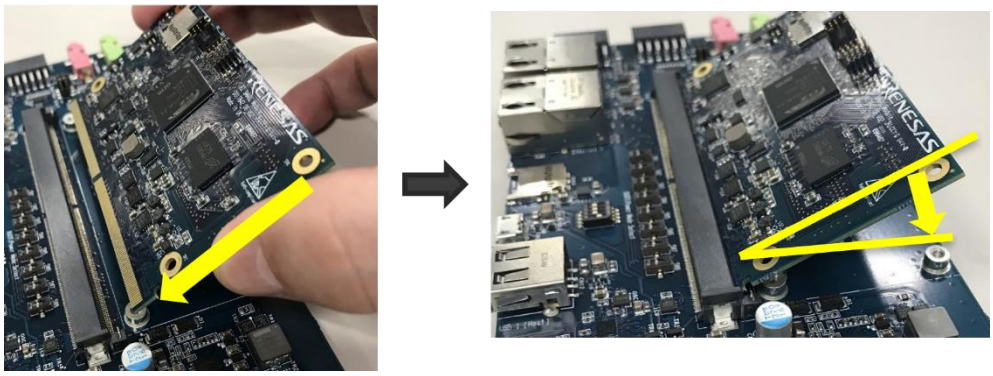


Specially shaped screw threads

2. Remove the screw and the board will stand up at an angle. Slide it out.

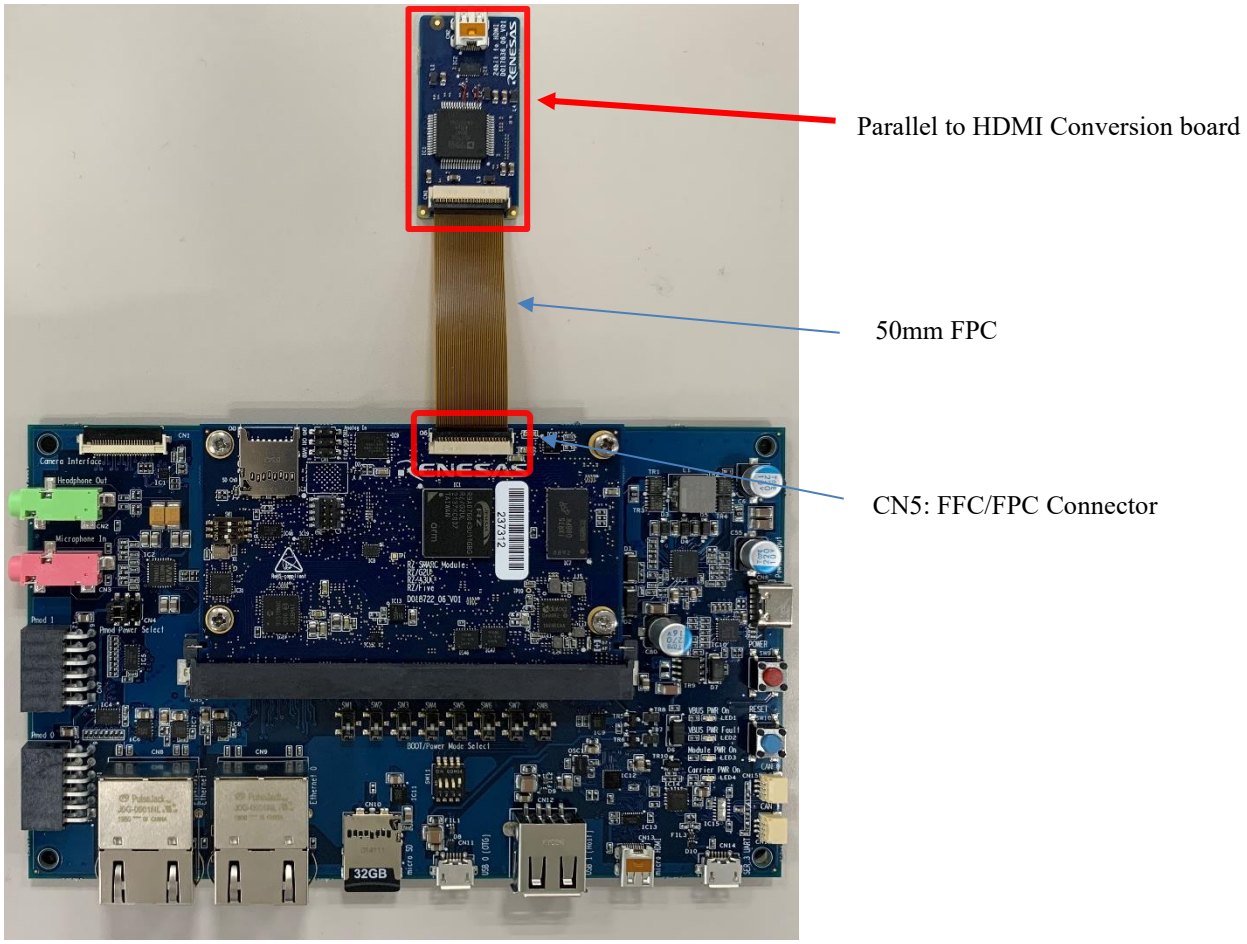


3. Insert the replacement into the board diagonally, then roll the SMARC board parallel to the board and fix it with screws.



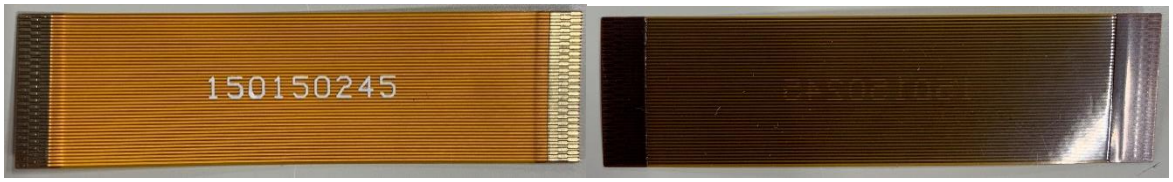
8.4 How to connect Parallel to HDMI Conversion board for RZ/G2UL Evaluation kit (smarc-rzg2ul)

Please connect CN1 of the Parallel to HDMI Conversion board to CN5 via the 50mm FPC (228-000071-01).



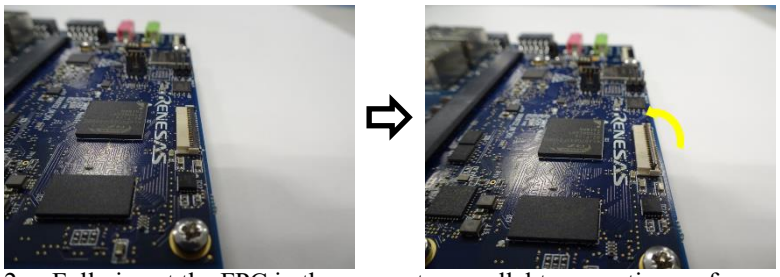
The termination procedure of the 50mm FPC is follows.

The top and bottom surfaces of the FPC are shown below.

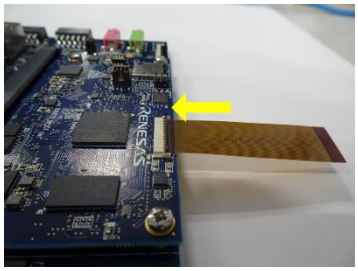


Top surface Bottom surface

1. Lift up the actuator. Use thumb or index finger.

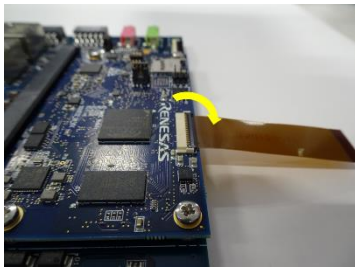


2. Fully insert the FPC in the connector parallel to mounting surface, with the exposed conductive traces facing down



3. Rotate down the actuator until firmly closed.

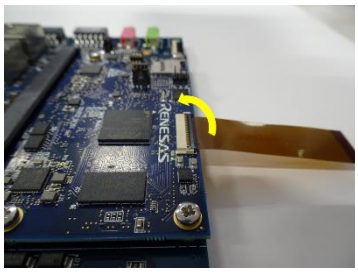
Note: The PFC must be fully inserted in the connector. If not fully inserted, the actuator will not close property. Should this be the case, lift up the actuator and repeat the process (starting with Step 1 above)



4. FPC Removal

1) Lift up the actuator.

2) Carefully remove the FPC



8.5 How to boot from eMMC

In this section, the steps to boot from eMMC to RZ/G2L and RZ/G2LC are described.

8.5.1 Rebuild rootfs

Build a rootfs by following section 2 Build Instructions. After that, please rebuild the rootfs with the command below.

```
$ cd ~/rz_vlp_v${PACKAGE_VERSION}
$ source poky/oe-init-build-env
$ echo 'IMAGE_INSTALL_append = " e2fsprogs-mke2fs"'>> conf/local.conf
$ bitbake core-image-xxxxx (ex. core-image-weston)
```

8.5.2 Writing Bootloader for eMMC Boot

For the boot operation, EXT_CSD register of eMMC needs to be modified and two boot loader files need to be written to the target board. Modifying register address and value information are described in **Table 13**, and corresponding bootloader files and specified address information are depending on each target board as described in **Table 14**.

After booting the Flash Writer (Refer to section 4.3), “EM_SECSD” command of Flash Writer is used to modify EXT_CSD register of eMMC to enable eMMC boot.

Then, “EM_W” command of Flash Writer is used to write boot loader binary files. This command receives binary data from the serial port and writes the data to a specified address of the eMMC with information where the data should be loaded on the address of the main memory.

For example, this part describes how to modify EXT_CSD register and write boot loader files in the case of RZ/G2L Evaluation Board Kit:

```
>EM_SECSD
  Please Input EXT_CSD Index(H'00 - H'1FF) :b1
  EXT_CSD[B1] = 0x00
  Please Input Value(H'00 - H'FF) :2
  EXT_CSD[B1] = 0x02
>EM_SECSD
  Please Input EXT_CSD Index(H'00 - H'1FF) :b3
  EXT_CSD[B3] = 0x00
  Please Input Value(H'00 - H'FF) :8
  EXT_CSD[B3] = 0x08
```

```
>EM_W
EM_W Start --------------
-----------------------------------------------------------
Please select,eMMC Partition Area.
 0:User Partition Area    : 62160896 KBytes
  eMMC Sector Cnt : H'0 - H'0768FFFF
 1:Boot Partition 1       : 32256 KBytes
  eMMC Sector Cnt : H'0 - H'0000FBFF
 2:Boot Partition 2       : 32256 KBytes
  eMMC Sector Cnt : H'0 - H'0000FBFF
-----------------------------------------------------------
  Select area(0-2)>1
-- Boot Partition 1 Program ----------------------------
Please Input Start Address in sector :1
Please Input Program Start Address : 11e00
Work RAM(H'50000000-H'50FFFFFF) Clear....
please send ! ('.' & CR stop load)
```

Send the data of “bl2_bp_spi-smarc-rzg2l_pmic.srec” from terminal software after the message “please send !” is shown.

After successfully downloading the binary, messages like below are shown on the terminal.

```
SAVE -FLASH.......
EM_W Complete!
```

Next, write another loader file by using EM_W command again.

```
> EM_W
EM_W Start --------------
-----------------------------------------------------------
Please select,eMMC Partition Area.
 0:User Partition Area   : 62160896 KBytes
  eMMC Sector Cnt : H'0 - H'0768FFFF
 1:Boot Partition 1      : 32256 KBytes
  eMMC Sector Cnt : H'0 - H'0000FBFF
 2:Boot Partition 2      : 32256 KBytes
  eMMC Sector Cnt : H'0 - H'0000FBFF
-----------------------------------------------------------
  Select area(0-2)>1
-- Boot Partition 1 Program ----------------------------
Please Input Start Address in sector :100
Please Input Program Start Address : 0
Work RAM(H'50000000-H'50FFFFFF) Clear....
please send ! ('.' & CR stop load)
```

Send the data of “fip-smarc-rzg2l_pmic.srec” from terminal software after the message “please send !” is shown.

After successfully downloading the binary, messages like below are shown on the terminal.

```
SAVE -FLASH.......
EM_W Complete!
```

After writing two loader files normally, turn off the power of the board by changing the SW11.

Table 13. Address of EXT_CSD register of eMMC for eMMC boot

Address	Value to write
0xB1	0x02
0xB3	0x08

Table 14. Address for sending each loader binary file for eMMC boot

RZ/G2L Evaluation Board Kit

File name	Partition to save to eMMC	Address to save to eMMC	Address to load to RAM
bl2_bp_sip-smarc-rzg2l_pmic.srec	1	00000001	11E00
fip-smarc-rzg2l_pmic.srec	1	00000100	00000

RZ/G2LC Evaluation Board Kit

File name	Partition to save to eMMC	Address to save to eMMC	Address to load to RAM
bl2_bp-smarc-rzg2lc.srec	1	00000001	11E00
fip-smarc-rzg2lc.srec	1	00000100	00000

8.5.3 Create a microSD card to boot linux for eMMC boot

Please create a microSD card by following the instructions in section 3 Preparing the SD Card. After that, please return to the following instructions before unmounting the micro SD card.

Copy the kernel image, device tree file, and rootfs to the second partition of the microSD card.

```
$ cd /media/user/rootfs/home/root/
$ sudo cp $WORK/build/tmp/deploy/images/<board>/<Linux kernel> ./
$ sudo cp $WORK/build/tmp/deploy/images/<board>/<Devise tree> ./
$ sudo cp $WORK/build/tmp/deploy/images/<board>/<root filesystem> ./
$ cd $WORK
$ sudo umount /media/user/rootfs
```

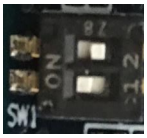
8.5.4 Writing rootfs to eMMC

To set the board to eMMC Boot mode, set the SW11 as below:

Table 15. SW11

1	2	3	4
ON	OFF	OFF	ON

Note) Set the SW1 on SoM module to eMMC mode. Please refer to the section 4.1.2. For RZ/G2LC, please refer to User's Manual: Hardware (RZ/G2LC-EVKIT - Evaluation Board Kit for RZ/G2LC MPU | Renesas).



Turn on the board by pressing the reset button SW10. After Linux boots, please log in as root and create partitions on the eMMC by running the following commands.

```
root@smarc-rzg2l:~# fdisk /dev/mmcblk0

Welcome to fdisk (util-linux 2.35.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Command (m for help): o
Created a new DOS disklabel with disk identifier 0xf3d53104.

Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): (Push the enter key)
Partition number (1-4, default 1): (Push the enter key)
First sector (2048-124321791, default 2048): (Push the enter key)
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-124321791, default 124321791): +50
0M

Created a new partition 1 of type 'Linux' and of size 500 MiB.

Command (m for help): n
Partition type
   p   primary (1 primary, 0 extended, 3 free)
   e   extended (container for logical partitions)
Select (default p): (Push the enter key)

Using default response p.
Partition number (2-4, default 2): (Push the enter key)
First sector (1026048-124321791, default 1026048): (Push the enter key)
```

```
Last sector, +/-sectors or +/-size{K,M,G,T,P} (1026048-124321791, default 124321791):
(Push the enter key)

Created a new partition 2 of type 'Linux' and of size 58.8 GiB.

Command (m for help): p
Disk /dev/mmcblk0: 59.29 GiB, 63652757504 bytes, 124321792 sectors
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: 0xf3d53104

Device         Boot   Start       End   Sectors  Size Id Type
/dev/mmcblk0p1         2048   1026047   1024000  500M 83 Linux
/dev/mmcblk0p2      1026048 124321791 123295744 58.8G 83 Linux

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.

root@smarc-rzg2l:~#
```

Format eMMC.

```
root@smarc-rzg2l:~# mkfs.ext4 /dev/mmcblk0p1
root@smarc-rzg2l:~# mkfs.ext4 /dev/mmcblk0p2
```

Format the eMMC and write the kernel, device tree, and rootfs.

```
root@smarc-rzg2l:~# mount /dev/mmcblk0p1 /mnt/
root@smarc-rzg2l:~# cp Image-smarc-rzg2l.bin /mnt/
root@smarc-rzg2l:~# cp Image-r9a07g054l2-smarc.dtb /mnt/
root@smarc-rzg2l:~# umount /dev/mmcblk0p1
root@smarc-rzg2l:~# mount /dev/mmcblk0p2 /mnt/
root@smarc-rzg2l:~# tar xf /home/root/core-image-weston-smarc-rzg2l.tar.bz2 \
-C /mnt/
root@smarc-rzg2l:~# umount /dev/mmcblk0p2
```

8.5.5 Setting U-boot for eMMC boot

Reset the board by pressing the reset button SW10 and interrupt the boot process by pressing the Enter key.

```
U-Boot 2024.07 (Sep 22 2025 - 10:51:32 +0000)

CPU:   Renesas Electronics CPU rev 1.0
Model: smarc-rzg2l
DRAM:  1.9 GiB
SF: Detected mt25qu512a with page size 256 Bytes, erase size 4 KiB, total 64 MiB
Core:  33 devices, 19 uclasses, devicetree: fit
WDT:   watchdog@0000000012800800
WDT:   Started watchdog@12800800 with servicing every 1000ms (60s timeout)
MMC:   sd@11c00000: 0, sd@11c10000: 1
Loading Environment from MMC... Reading from MMC(0)... OK
In:    serial@1004b800
Out:   serial@1004b800
Err:   serial@1004b800
WDT:   Started watchdog@12800800 with servicing every 1000ms (60s timeout)
U-boot WDT started!
Net:   eth0: ethernet@11c20000
Warning: ethernet@11c30000 (eth1) using random MAC address - fa:72:9e:48:2b:2d
, eth1: ethernet@11c30000Hit any key to stop autoboot:  0
=>
```

Set environment variables in the U-Boot environment to boot from eMMC. The following variables are for the RZ/G2L board. When using other boards, please replace the file names in "bootcmd" and set the IP address of your Linux host PC.

```
=> setenv bootargs 'root=/dev/mmcblk0p2 rootwait'
=> setenv bootcmd 'mmc dev 1; ext4load mmc 0:1 0x48080000 Image-smarc-rzg2l.bin; ext4l
oad mmc 0:1 0x48000000 Image-r9a07g044l2-smarc.dtb; booti 0x48080000 - 0x48000000'
=> saveenv
Saving Environment to MMC... Writing to MMC(0)….OK
```

Please reset the board again for eMMC boot.

8.6 How to boot from eSD

In this section, the steps to boot from eSD on RZ/G2L are described as an example.

RZ/G2L can also be booted from a micro-SD card, this is booting mode called "Booting from eSD" in the User Manual.

8.6.1 Prepare micro SD card

Prepare the micro SD card using the wic image file.

Two files (bl2_bp_esd-smarc-<device>.bin and fip-smarc-<device>.bin) are used for boot from eSD.

Table 16. File and directory in the micro SD card

Type/Number	Filesystem	Contents
Primary #1	FAT32	Flash_Writer_SCIF_<device, memory size>.mot bl2_bp_mmc-smarc-<device>.srec bl2_bp_spi-smarc-<device>.srec bl2_bp_esd-smarc-<device>.bin fip-smarc-<device>.srec fip-smarc-<device>.bin
Primary #2	Ext4	./ ├── bin ├── boot │ ├── Image │ └── <device>-smarc.dtb ├── dev ├── etc ├── home ├── lib ├── media ├── mnt ├── proc ├── run ├── sbin ├── sys ├── tmp ├── usr └── var

8.6.2 Set SMARC EVK board for eSD boot

Change the switches to eSD boot on the carrier board acting on SW11 (SW11-1 ON, SW11-2 ON, SW11-3 OFF, SW11-4 ON).

Change SW1 on the module to select micro SD card slot (SW1-1 ON, SW1-2 ON) for RZ/G2L EVK:

For RZ/G2LC and RZ/G2UL EVK, please refer to User's Manual: Hardware (RZ/G2LC-EVKIT - Evaluation Board Kit for RZ/G2LC MPU | Renesas, RZ/G2UL-EVKIT - Evaluation Board Kit for RZ/G2UL MPU | Renesas).

Please insert the micro SD card into the SOM module slot.

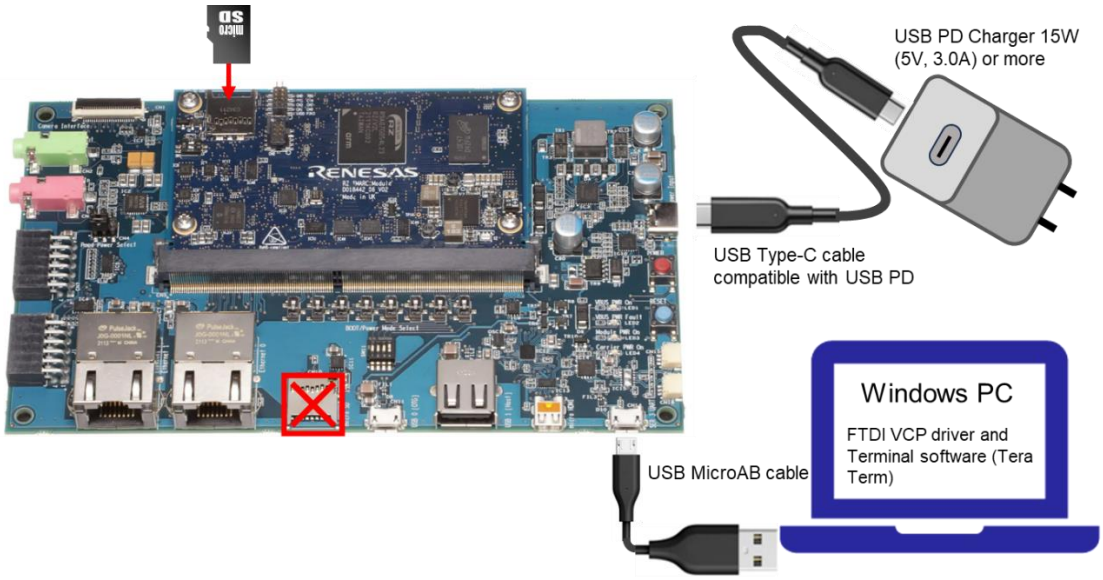


Figure 8. Operating environment for eSD boot

8.6.3 Power on and boot

After pressing the POWER button (SW9) to turn on the power and RESET button (SW10), Linux will be booted from eSD.

```
NOTICE:  BL2: v2.10.5(release):2.10.5/rz_1.1.0_rc1-dirty
NOTICE:  BL2: Built : 14:24:16, Aug  6 2025
NOTICE:  BL2: RZ/G2L
NOTICE:  BL2: SYS_LSI_MODE: 0x0
NOTICE:  BL2: SYS_LSI_DEVID: 0x4841c447
NOTICE:  BL2: SYS_LSI_PRR: 0x0
NOTICE:  BL2: Booting BL31
NOTICE:  BL31: v2.10.5(release):2.10.5/rz_1.1.0_rc1-dirty
NOTICE:  BL31: Built : 14:24:16, Aug  6 2025


U-Boot 2024.07 (Sep 10 2025 - 18:07:39 +0000)

CPU:   Renesas Electronics CPU rev 1.0
Model: smarc-rzg2l
DRAM:  1.9 GiB
SF: Detected mt25qu512a with page size 256 Bytes, erase size 4 KiB, total 64 MiB
Failed to send NVCR Read command: -38
Core:  33 devices, 19 uclasses, devicetree: separate
WDT:   watchdog@0000000012800800
WDT:   Started watchdog@12800800 with servicing every 1000ms (60s timeout)
MMC:   sd@11c00000: 0, sd@11c10000: 1
Loading Environment from MMC... Reading from MMC(0)... *** Warning - bad CRC, using de
fault environment

In:    serial@1004b800
Out:   serial@1004b800
Err:   serial@1004b800
WDT:   Started watchdog@12800800 with servicing every 1000ms (0s timeout)
U-boot WDT started!
Net:
Warning: ethernet@11c20000 (eth0) using random MAC address - 3a:f9:ae:21:16:6d
eth0: ethernet@11c20000
Warning: ethernet@11c30000 (eth1) using random MAC address - 0e:be:43:ff:6d:3e
, eth1: ethernet@11c30000
Hit any key to stop autoboot:  0## Resetting to default environment
Card did not respond to voltage select! : -110
20070912 bytes read in 1655 ms (11.6 MiB/s)
39353 bytes read in 6 ms (6.3 MiB/s)
Error: Bad gzipped data
Moving Image from 0x48080000 to 0x48200000, end=495a0000
## Flattened Device Tree blob at 48000000
   Booting using the fdt blob at 0x48000000
   Loading Device Tree to 0000000057ff3000, end 0000000057fff9b8 ... OK
Starting kernel ...

[    0.000000] Booting Linux on physical CPU 0x0000000000 [0x412fd050]
[    0.000000] Linux version 6.10.184-cip36-yocto-standard (oe-user@oe-host) (aarc
 :
 :
Poky (Yocto Project Reference Distro) 3.1.33 smarc-rzg2l ttySC0

BSP: RZG2L/RZG2L-SMARC-EVK/4.0.0
LSI: RZG2L
```

```
Version: 4.0.0
smarc-rzg2l login: root
root@smarc-rzg2l:~#
```

8.7 Docker

This section explains how to build the VLP enabling Docker and how to obtain and run the "hello-world" image.

(1) Add layers to enable Docker

After completing the step (4) in section 2.2, run the following commands. Once executed, proceed until step (7) in section 2.2 to build the VLP enabling Docker.

```
$ cd ~/rzg_vlp_v${PACKAGE_VERSION}/build
$ bitbake-layers add-layer ../meta-openembedded/meta-filesystems
$ bitbake-layers add-layer ../meta-openembedded/meta-networking
$ bitbake-layers add-layer ../meta-virtualization
```

(2) Boot the evaluation board

Follow sections 4 and 5, boot the evaluation board, and proceed to the next step in this section.

(3) Check Docker on the evaluation board (Optional)

After booting the board, optionally run the following commands to display the status of the Docker Daemon and its version.

```
root@smarc-rzg2l:~# systemctl status docker.service
root@smarc-rzg2l:~# docker version
```

(4) Hello world

Run the commands below to get the docker image of hello-world and run the image.

```
root@smarc-rzg2l:~# docker pull hello-world
root@smarc-rzg2l:~# docker run hello-world
Hello from Docker!
```

8.8 Booting Setup with Ubuntu PC

This section describes how to write Flash writer and bootloaders using Ubuntu PC. Here is an example using an Ubuntu PC and minicom.

Please connect the reference board to your Ubuntu PC.

(1) Check the serial connected to the Ubuntu PC

```
$ ls -l /dev/serial/by-id
```

Assuming that the serial device is connected to "ttyUSB0", the following procedure is introduced.

(2) Allow access to the serial device

```
$ sudo chmod 666 /dev/ttyUSB0
```

(3) Get a minicom communication app

```
$ sudo apt-get install minicom
```

(4) Configure settings that can be executed by the logged-in user, and reboot

```
$ sudo usermod -a -G dialout $USER
$ sudo shutdown -r now
```

(5) Connect the minicom communication app to the serial device

```
$ minicom -D /dev/ttyUSB0
```

To change the settings, press "Ctrl+A" -> "Z" to display the HELP screen and select "Other Function(O)"-> "Serial port setup". Normal communication is now possible.

(6) Send a file

This section describes the case of rewriting U-boot.

```
SCIF Download mode (w/o verification)
(C) Renesas Electronics Corp.

-- Load Program to SystemRAM  -----------------
Please send !
```

“Ctrl + A” -> “Z”

```
+-----------------------------------------------------------------------+
|                       Minicom Command Summary                         |
|                                                                       |
|               Commands can be called by CTRL-A <key>                  |
|                                                                       |
|                Main Functions                  Other Functions        |
|                                                                       |
| Dialing directory..D  run script (Go)....G | Clear Screen.......C |
| Send files.........S  Receive files......R | cOnfigure Minicom..O |
| comm Parameters....P  Add linefeed.......A | Suspend minicom....J |
| Capture on/off.....L  Hangup.............H | eXit and reset.....X |
| send break.........F  initialize Modem...M | Quit with no reset.Q |
| Terminal settings..T  run Kermit.........K | Cursor key mode....I |
| lineWrap on/off....W  local Echo on/off..E | Help screen........Z |
```

```
| Paste file.........Y  Timestamp toggle...N | scroll Back........B |
| Add Carriage Ret...U                                             |
|                                                                  |
|             Select function or press Enter for none.             |
+------------------------------------------------------------------+
```

Please select Senf files “S” and ascii.

```
+-[Upload]--+
| zmodem    |
| ymodem    |
| xmodem    |
| kermit    |
| ascii     |
+-----------+--
```

First please select Flash Write file.

```
+-----------------------[Select a file for upload]-------------------------+
|Directory: /home/user                                                     |
| .sudo_as_admin_successful                                                |
| Flash_Writer_SCIF_RZG2L_SMARC_PMIC_DDR4_2GB_1PCS.mot                     |
| bl2_bp-smarc-rzg2l_pmic.srec                                             |
| fip-smarc-rzg2l_pmic.srec                                                |
|              ( Escape to exit, Space to tag )                            |
+--------------------------------------------------------------------------+

               [Goto]  [Prev]  [Show]   [Tag]  [Untag] [Okay]
```

Start uploading. Press any key when upload completed.

```
+------------[ascii upload - Press CTRL-C to quit]------------ ---------+
|ASCII upload of " Flash_Writer_SCIF_RZG2L_SMARC_PMIC_DDR4_2GB_1PCS.mot |
|                                                                       |
|82.5 Kbytes transferred at 11234 CPS... Done.                          |
|                                                                       |
| READY: press any key to continue...                                   |
|                                                                       |
+-----------------------------------------------------------------------+
```

```
-- Load Program to SystemRAM ---------------
please send !
>
```

Next is writing U-boot step. Please refer to 4.4 Write the Bootloader .Upload with the “ascii” command in the same way as a Flash Writer file.

8.9 Using kas tool to build BSP

kas provides an easy mechanism to set up and build Yocto BSP projects. For kas's user guide, how to install kas..., please refer to: https://kas.readthedocs.io/en/latest/userguide.html. For command-line usage and kas environment variables, please also refer to the user guide.

(1) Prepare the build environment

Please proceed the steps 2.2(1) and 2.2(2) to prepare the build environment. If you have already done the steps, please skip this step.

(2) Enable Graphics and Video Codec

The graphics package and the video codec package can be used at the same time. And one of the packages can be used.

Please check the website page below for available combination with Yocto recipe package.

RZ MPU Verified Linux Package [6.1-CIP] | Renesas

The following operations are the EN packages. If you use the JP package, replace EN with JP.

For RZ/G2L and RZ/G2LC

The graphics package and the video codec package can be used at the same time. And one of the packages can be used.

If you want to enable the Graphics on RZ/G2L and RZ/G2LC when building **core-image-weston**, please copy the Graphics package (RTK0EF0045Z14001ZJ-<version>_EN.zip or RTK0EF0045Z14001ZJ-<version>_JP.zip) to working directory and run the commands below. If you build core-image-minimal, please ignore this step.

```
$ cd ~/rz_vlp_v${PACKAGE_VERSION}/meta-renesas
$ unzip ../RTK0EF0045Z14001ZJ-<version>_EN.zip
$ tar zxvf ./RTK0EF0045Z14001ZJ-<version>_EN/meta-rz-features_graphics_<version>.\
tar.gz
```

For RZ/G2L

If you want to enable the video codec on RZ/G2L when building **core-image-weston**, please copy the video codec package (RTK0EF0045Z16001ZJ-<version>_EN.zip or RTK0EF0045Z16001ZJ-v<version>_JP.zip) to working directory and run the commands below.

```
$ cd ~/rz_vlp_v${PACKAGE_VERSION}/meta-renesas
$ unzip ../RTK0EF0045Z16001ZJ-<version>_EN.zip
$ tar zxvf ./RTK0EF0045Z16001ZJ-<version>_EN/meta-rz-features_codec_<version>.tar\
.gz
```

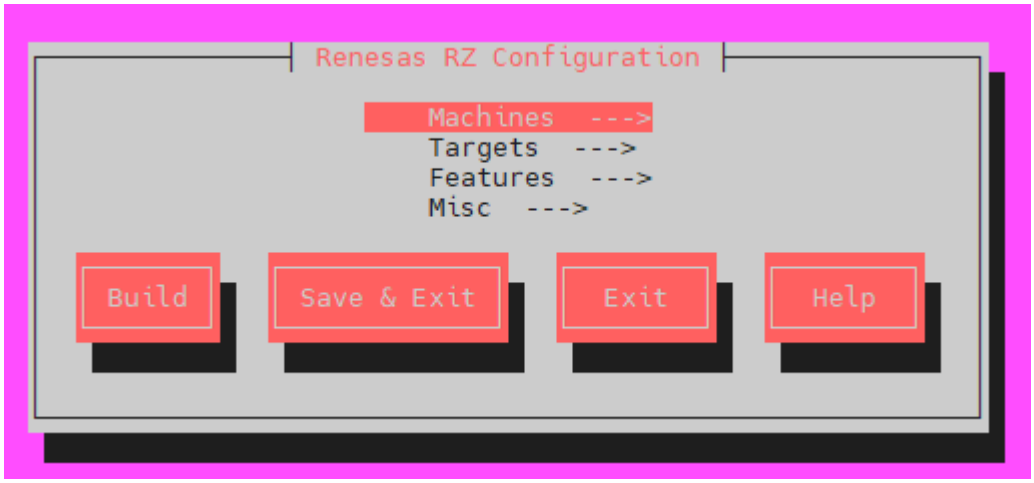
(3) Launch the kas menu and start the building

To build VLP with kas tool, run kas menu command target a Kconfig file in the folder meta-renesas.

```
$ cd ~/rz_vlp_v${PACKAGE_VERSION}/meta-renesas
$ git init
$ kas menu Kconfig
```

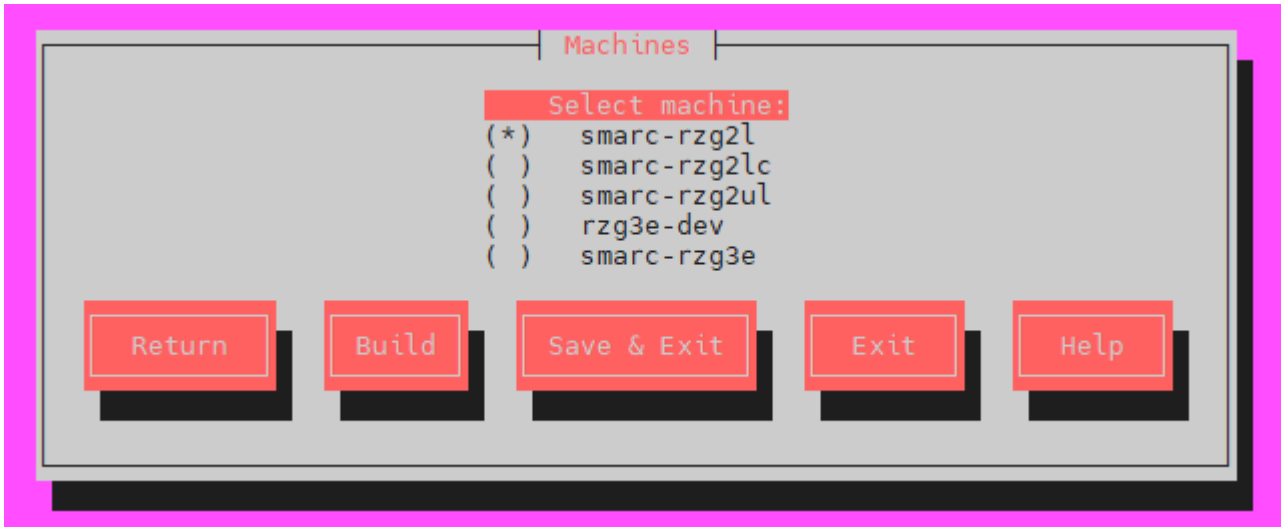
When the menu appears, continue to select the expected configuration (Machines, Targets and Features.).

- **Machines option**

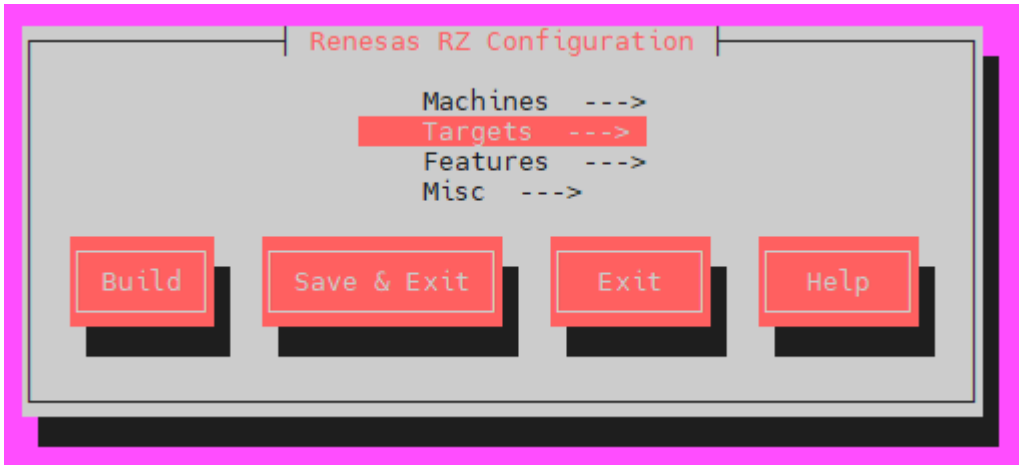


Machines can be selected as below:

- smarc-rzg2l
- smarc-rzg2lc

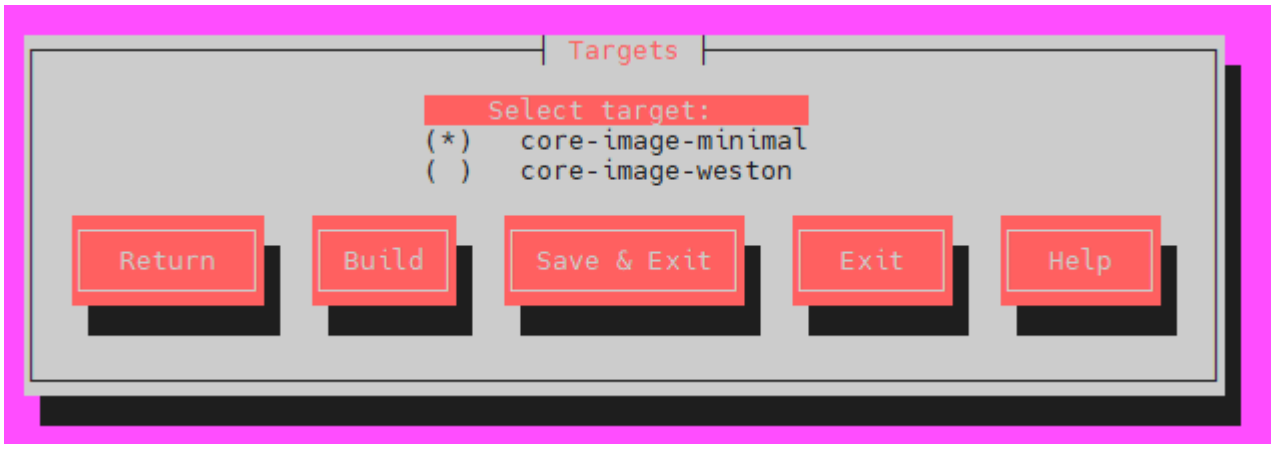


- **Targets option**

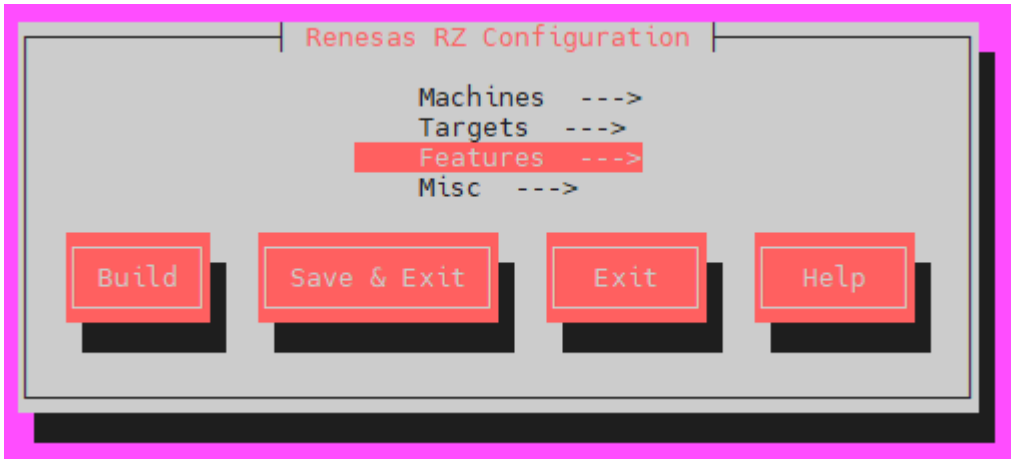


Targets can be selected as below:

- core-image-minimal
- core-image-weston

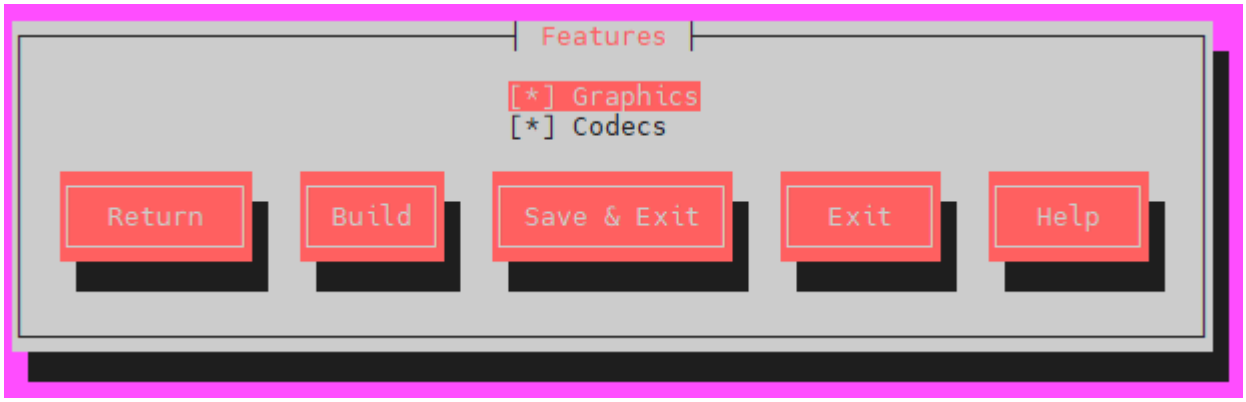


- **Features option**



Features can be selected as below:

- Graphics
- Codecs



Then push "Build" button to build the image.

All necessary files will be generated in "~/rz_vlp_v${PACKAGE_VERSION}/meta-renesas/build/tmp/deploy/images" directory.

8.10 Device drivers

The following drivers are supported: For detailed information on how to use it, please refer to these documents in the BSP manual set.

File	Description
RTK0EF0045Z9006AZJ-v4.0.x.zip	BSP Manual Set for RZ/G2L,RZ/G2LC and RZ/G2UL.

Note) "x" is the version of the file. Please refer to the latest one.

Table 17. Support device drivers

Device Driver	Documents
Kernel Core	r01us0468ej0xxx-rz_Kernel_Core_UME.pdf
Direct Memory Access Controller (DMAC)	r01us0480ej0xxx-rz_DMAC_UME.pdf
Multi-function Timer Unit 3a (MTU3a)	r01us0476ej0xxx-rz_MTU3a_UME.pdf
Port Output Enable 3 (POE3)	r01us0552ej0xxx-rz_POE3_UME.pdf
General PWM Timer (GPT)	r01us0484ej0xxx-rz_GPT_UME.pdf
Watchdog Timer (WDT)	r01us0479ej0xxx-rz_WDT_UME.pdf
Serial Communication Interface with FIFO A (SCIFA)	r01us0483ej0xxx-rz_SCIFA_UME.pdf
Renesas Serial Peripheral Interface (RSPI)	r01us0481ej0xxx-rz_RSPI_UME.pdf
SPI Multi IO Bus Controller	r01us0482ej0xxx-rz_SPI_Multi IO_UME.pdf
I2C Bus Interface	r01us0477ej0xxx-rz_I2C_UME.pdf
Serial Sound Interface	r01us0490ej0115-rz_Audio_UME.pdf
RS-CANFD Interface	r01us0478ej0xxx-rz_CANFD_UME.pdf
Gigabit Ethernet Interface	r01us0475ej0xxx_rz_Gigabit_Ethernet_UME.pdf
A/D Converter	r01us0487ej0xxx-rz_AD_Converter_UME.pdf
USB 2.0 Host	r01us0485ej0xxx-rz_USB2.0_Host_UME.pdf
USB 2.0 Function	r01us0491ej0xxx-rz_USB2.0_Function_UME.pdf
LCD Controller (LCDC)	r01us0489ej0xxx-rz_LCDC_UME.pdf
Camera Receiving Unit (CRU)	r01us0499ej0xxx-rz_CRU_UME.pdf
SD/MMC Host Interface	r01us0474ej0xxx-rz_SD_MMC_UME.pdf
GPIO	r01us0488ej0xxx-rz_GPIO_UME.pdf
Power Management	r01us0605ej0xxx-rz_Power_Management_UME.pdf
Thermal Sensor Unit (TSU)	r01us0486ej0xxx-rz_Thermal_Sensor_UME.pdf
Error Correction Code (ECC)	r01us0647ej0xxx-rz_ECC_UME.pdf

Note) "xxx" is the revision of the file. Please refer to the latest one.

9. Revision History

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	May 30, 2025	-	First edition issued.
1.01	Nov 30, 2025	-	- Add RZ/G2UL support information
		24	- Update QSPI Save address for the latest TF-A
		37	- Add appendix “8.4 How to connect Parallel to HDMI Conversion board for RZ/G2UL Evaluation kit (smarc-rzg2ul)”

Website and Support

Renesas Electronics Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/contact/