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

R01US0616EJ0107
Rev.1.07
Jul. 31, 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

RZ/G2L reference board

- RZ/G2L Evaluation Board Kit PMIC version (smarc-rzg2l-pmic) (*1)
  - RZ/G2L SMARC Module Board (P/N: RTK9744L23C01000BE)
  - RZ SMARC Series Carrier Board (P/N: RTK97X4XXXB00000BE)

RZ/G2LC reference board

- RZ/G2LC Evaluation Board Kit (smarc-rzg2lc) (*2)
  - RZ/G2LC SMARC Module Board (P/N: RTK9744C22C01000BE)
  - RZ SMARC Series Carrier Board (P/N: RTK97X4XXXB00000BE)

RZ/G2UL reference board

- RZ/G2UL Evaluation board Kit (smarc-rzg2ul) (*3)
  - RZ/G2UL SMARC Module Board (P/N: RTK9743U11C01000BE)
  - RZ SMARC Series Carrier Board (P/N: RTK97X4XXXB00000BE)
  - Parallel to HDMI Conversion board (This board is included with RTK9743U11C01000BE and is not sold separately, so a part number does not exist.)

🚧 Please refer to Appendix 2.2 and 2.3 for how to distinguish and replace each board.
🚧 Please refer to Appendix 2.4 for how to connect the Parallel to HDMI Conversion board for RZ/G2UL Evaluation board Kit (smarc-rzg2ul).

(*1) “RZ/G2L Evaluation Board Kit PMIC” includes the RZ/G2L SMARC Module Board and the RZ SMARC Series Carrier Board.
(*2) “RZ/G2LC Evaluation Board Kit” includes the RZ/G2LC SMARC Module Board and the RZ SMARC Series Carrier Board.
(*3) “RZ/G2UL Evaluation board Kit” includes the RZ/G2UL SMARC Module Board and the RZ SMARC Series Carrier Board.
The “Evaluation board Kit for RZ/G2UL MPU” will be called “RZ/G2UL Evaluation Kit” in the next section.

Target Software

- RZ/G Verified Linux Package version 3.0.7 or later. (hereinafter referred to as “VLP/G”)

Contents

1. Environment Requirement

The environment for building the Verified Linux Package (hereinafter referred to as “VLP”) is listed in **Table 1**. Please refer to the documents below for details about setting up the environment:

Figure 1-1 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.

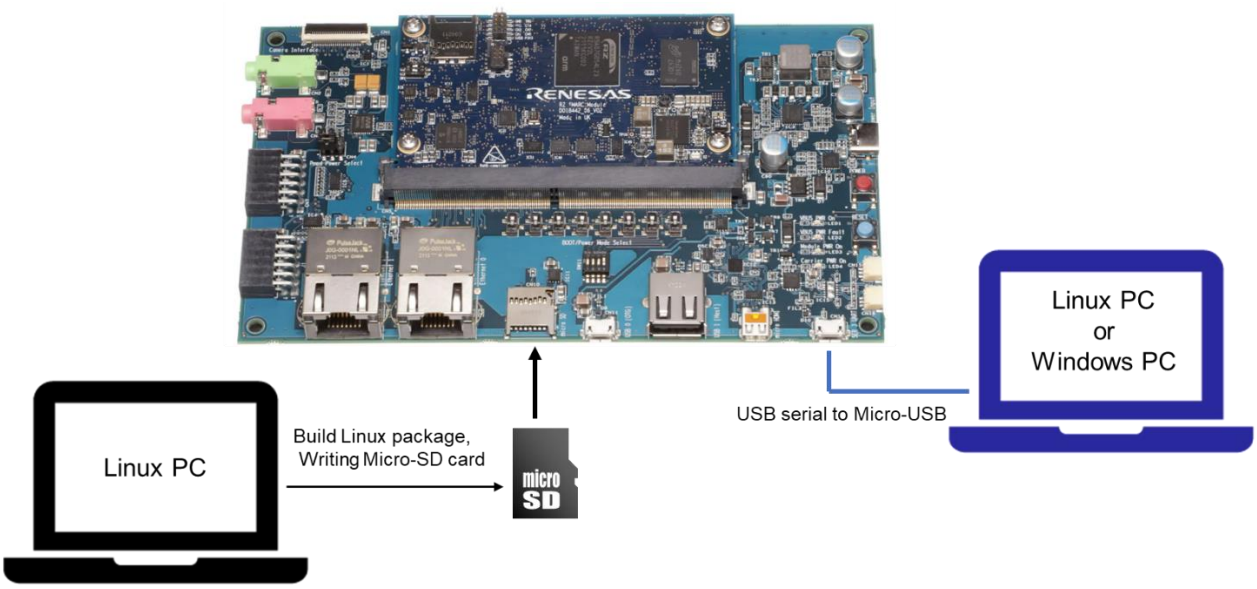


Figure 1. Recommend environment.

Note: The board shown in Figure 1 is RZ/V2L, but RZ/G2L has the same structure.

Table 1. Equipment and Software for Developing Environments of RZ/G Linux Platform

Equipment		Description
Linux Host PC		Used as build/debug environment 100GB free space on HDD or SSD is necessary
	OS	Ubuntu 22.04 LTS 64 bit OS must be used. 22.04 inside a docker container also OK.
Windows Host PC		Used as debug environment, controlling with terminal software
	OS	Windows 10 or 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/G supports can be built on an “offline” environment.
The word “offline” means an isolated environment which does not connect to any network. Since VLP/G includes all necessary source codes of OSS except for the Linux kernel, VLP/G 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-bsp
- core-image-weston (including the SDK build)
- core-image-qt (including the SDK build)

2. Build Instructions

2.1 Required Host OS

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

Ubuntu 22.04 is required to build the VLP/G. This is because it was the only host operating system tested and is a specific requirement for Yocto 3.1 (dunfell). Using Ubuntu 24.04 is not supported.

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-get install gawk wget git-core diffstat unzip texinfo gcc-multilib \
build-essential chrpath socat cpio python3 python3-pip python3-pexpect xz-utils \
debianutils iputils-ping libsdl1.2-dev xterm p7zip-full libyaml-dev libssl-dev \
bmap-tools
```

Please refer to the URL below for detailed information:

- https://docs.yoctoproject.org/3.1.33/brief-yoctoprojectqs/brief-yoctoprojectqs.html

Run the commands below and set the username 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) 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.

```
$ PACKAGE_VERSION=v3.0.7
$ mkdir ~/rzg_vlp_${PACKAGE_VERSION}
$ cd ~/rzg_vlp_${PACKAGE_VERSION}
$ cp ../<package download directory>/*.zip .
$ unzip ./RTK0EF0045Z0021AZJ-${PACKAGE_VERSION}.zip
$ tar zxvf ./RTK0EF0045Z0021AZJ-${PACKAGE_VERSION}/rzg_vlp_${PACKAGE_VERSION}.tar.gz
```

Note) Please note that your building environment must have 100GB of free hard drive space in order 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.

Note) `${PACKAGE_VERSION}`: e.g v3.0.7

(2) Enable Graphics and Video Codec

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

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

RZ/G Verified Linux Package [5.10-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 also, 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.

Note: GRAPHICS_VERSION=v3.1.2.3 is **just an example** for **VLP/G 3.0.7**.

```
$ GRAPHICS_VERSION=v3.1.2.3
$ unzip ./RTK0EF0045Z14001ZJ-${GRAPHICS_VERSION}_EN.zip
$ tar zxvf ./RTK0EF0045Z14001ZJ-${GRAPHICS_VERSION}_EN/meta-rz-features_\
graphics_${GRAPHICS_VERSION}.tar.gz
```

For RZ/G2L

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

Note: CODEC_VERSION=v3.1.3.0 is **just an example** for **VLP/G 3.0.7**.

```
$ CODEC_VERSION=v3.1.3.0
$ unzip ./RTK0EF0045Z16001ZJ-${CODEC_VERSION}_EN.zip
$ tar zxvf ./RTK0EF0045Z16001ZJ-${CODEC_VERSION}_EN/meta-rz-features_\
codec_${CODEC_VERSION}.tar.gz
```

(3) 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-rzg2l/docs/template/conf/ source \
poky/oe-init-build-env build
```

(4) 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
```

- **Qt**: Please run the command below if you want to include Qt.

```
$ bitbake-layers add-layer ../meta-qt5
$ bitbake-layers add-layer ../meta-rz-features/meta-rz-graphics
$ 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-filesystems
$ bitbake-layers add-layer ../meta-openembedded/meta-networking
$ bitbake-layers add-layer ../meta-virtualization
```

(5) Decompress OSS files to "build" directory (Optional)

Run the commands below. This step is not mandatory and able to go to the step (6) in case the "offline" environment is not required. All OSS packages will be decompressed with this '7z' command.

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

Note) If this step is omitted and BB_NO_NETWORK is set to "0" in next step, all source codes will be downloaded from the repositories of each OSS via the internet when running bitbake command. Please note that if you do not use an "offline" environment, a build may fail due to the implicit changes of the repositories of OSS.

Open source software packages contain all source codes of OSSs. These are the same versions of OSSs used when VLP/G was verified.
If you are just evaluating VLP/G and RZ/G2L group, open source software packages are not mandatory to use. Usually, all the software can be built without using these files if your build machine is connected to the Internet.

Open source software packages are required for an "offline" environment. The word "offline" means an isolated environment which does not connect to any network. VLP/G can always build images in this "offline" environment by using these packages without affected from changes of original repositories of OSSs. Also, this "offline" environment always reproduces the same images as the images which were verified by Renesas. Note that if you build without using open source software packages, there are possibilities to use different source codes than Renesas used due to the implicit changes of the repositories of OSSs.

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

```
BB_NO_NETWORK = "1"
```

To change BB_NO_NETWORK from "0" to "1".

(6) 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 the **Table 2**.

Table 2. List of platforms and boards

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

<image name> can be selected below. Please refer to the **Table 3** for supported image details.

- core-image-minimal
- core-image-bsp
- core-image-weston
- core-image-qt

Table 3. Supported images of VLP/G

Image name	Target devices	Purpose
core-image-minimal	RZ/G2L, RZ/G2LC, RZ/G2UL	Minimal set of components
core-image-bsp	RZ/G2L, RZ/G2LC, RZ/G2UL	Minimal set of components plus audio support and some useful tools
core-image-weston	RZ/G2L, RZ/G2LC	Standard image with graphics support
core-image-qt	RZ/G2L, RZ/G2LC	Enable Qt LGPL version

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 4** will be generated by the bitbake command and will be in the **`build/tmp/deploy/images`** directory.

Table 4. Image files for RZ/G2L, RZ/G2LC, and RZ/G2UL

RZ/G2L PMIC ver	**Linux kernel**	Image-smarc-rzg2l.bin
	Device tree file	Image-r9a07g044l2-smarc.dtb
	root filesystem	*<image name>*-smarc-rzg2l.tar.bz2
	Boot loader	・bl2_bp-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.wic.gz *<image name>* -smarc-rzg2l.wic.bmap
RZ/G2LC	**Linux kernel**	Image-smarc-rzg2lc.bin
	Device tree file	Image-r9a07g044c2-smarc.dtb
	root filesystem	*<image name>*-smarc-rzg2lc.tar.bz2
	Boot loader	・bl2_bp-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.wic.gz *<image name>* -smarc-rzg2lc.wic.bmap
RZ/G2UL	**Linux kernel**	Image-smarc-rzg2ul.bin
	Device tree file	Image-r9a07g043u11-smarc.dtb
	root filesystem	*<image name>*-smarc-rzg2ul.tar.bz2
	Boot loader	・bl2_bp-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.wic.gz *<image name>* -smarc-rzg2ul.wic.bmap

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

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 ~/rzg_vlp_${PACKAGE_VERSION}/meta-renesas
$ patch -p1 < ../extra/0002-trusted-firmware-a-add-rd-wr-64-bit-reg-workaround.patch
$ patch -p1 < ../extra/0003-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) GPLv3 packages

In this release, the GPLv3 packages are disabled as default in *build/conf/local.conf*:

```
INCOMPATIBLE_LICENSE = "GPLv3 GPLv3+"
```

If you want to use GPLv3, just hide this line:

```
#INCOMPATIBLE_LICENSE = "GPLv3 GPLv3+"
```

If you want to change this setting after completing step (6) of section 2.2, create a new working directory and prepare the new building environment. Note that not doing this may cause a build error.

(3) Disable libraries of Graphics and Video Codec

If you want to disable the functions of the libraries of the graphics and the video codec, please add the following lines in `build/conf/local.conf`:

- Disable OpenGL ES library in the graphics package (*1)

```
DISTRO_FEATURES_remove = " opengles"
```

- Disable OpenCL library in the graphics package (*1)

```
DISTRO_FEATURES_remove = " opencl"
```

- Disable OpenMAX library for decode in the video codec package (*2)

```
DISTRO_FEATURES_remove = " hwh264dec hwh265dec"
```

- Disable OpenMAX library for encoding in the video codec package (*2)

```
DISTRO_FEATURES_remove = " hwh264enc"
```

(*1) This library is included in RTK0EF0045Z14001ZJ-<version>_EN.zip and RTK0EF0045Z14001ZJ-<version>_JP.zip
(*2) This library is included in RTK0EF0045Z16001ZJ-<version>_EN.zip and RTK0EF0045Z16001ZJ-<version>_JP.zip

(4) Real time performance

If you want to use the kernel which improves the real-time performance, please add the line below to the file "`~/rzg_vlp_v3.0.x/build/conf/local.conf`".

```
IS_RT_BSP="1"
```

(5) USB Video Class

USB Video Class (UVC) driver is not installed with the default settings of VLP/G due to its large size.

In case UVC devices such as USB cameras are necessary, please install the driver by adding the line below to `local.conf`.

```
IMAGE_INSTALL_append = " kernel-module-uvcvideo "
```

(6) CIP Core Packages

VLP/G includes Debian 10 (Buster), and Debian 11 (Bullseye) based CIP Core Packages and Buster is enabled by the default settings. These packages can be replaced with other versions of packages.

Note that network access is required to start the build process when you enable these packages except for Buster (or Bullseye) which is set as the default setting.
If you want to change this setting after completing step (6) of section 2.2, create a new working directory and prepare the new building environment. Note that not doing this may cause a build error.

CIP Core Packages are going to be maintained by the Civil Infrastructure Platform project. For more technical information, please contact Renesas.

1. **Buster** (default)**:**

   The following lines are added as default in the `local.conf`:

```
# Select CIP Core packages by switching between Buster and Bullseye.
# - Buster (default) : build all supported Debian 10 Buster recipes
# - Bullseye : build all supported Debian 11 Bullseye recipes
# - Not set (or different with above): not use CIP Core, use default packages version
in Yocto

CIP_MODE = "Buster"
```

2. **Bullseye:**

   Please change "`CIP_MODE`" in the `local.conf` to change from Buster to Bullseye:

```
# Select CIP Core packages by switching between Buster and Bullseye.
# - Buster (default) : build all supported Debian 10 Buster recipes
# - Bullseye : build all supported Debian 11 Bullseye recipes
# - Not set (or different with above): not use CIP Core, use default packages version
in Yocto

CIP_MODE = "Bullseye"
```

3. **No CIP Core Packages:**

   If the CIP Core Packages are unnecessary, comment out and add the following lines to disable CIP CORE Packages in `local.conf`:

```
# Select CIP Core packages by switching between Buster and Bullseye.
# - Buster (default) : build all supported Debian 10 Buster recipes
# - Bullseye : build all supported Debian 11 Bullseye recipes
# - Not set (or different with above): not use CIP Core, use default packages version
in Yocto

#CIP_MODE = "Buster"
```

Note) The above 3 settings disable GPLv3 packages as default. In case the GPLv3 packages are required, please comment out the following line in the `local.conf`.

```
# INCOMPATIBLE_LICENSE = "GPLv3 GPLv3+"
```

By building the VLP, the packages will be replaced as below in the **Table 5**.

Table 5. Versions of all Buster and Bullseye Debian packages

Package	Buster Debian	Bullseye Debian
attr	2.4.48	2.4.48
busybox	1.30.1	1.30.1
coreutils	6.9	-
gcc	8.3.0	9.5.0
glib-2.0	2.58.3	2.62.6
glibc	2.28	2.31
gnupg	1.4.7	-
kbd	2.0.4	-
libassuan0	2.5.2	2.5.3
libgcrypt	1.8.4	-
libunistring	0.9.10	0.9.10
libnss	0.14.1	-
openssh	7.9p1	-
perl	5.30.1	-
pkgconfig	0.29	-
quilt	0.65	-

Note)
(-) These packages are not supported with Bullseye Debian version, so they used No CIP CORE version.

(7) ECC

If you want to use ECC, see r01us0647ej0100-rz-mpu_ECC_UME.pdf included in the BSP manual set on the Renesas Web. Please also check section 8.6.

(8) WIC image

The name “WIC” is derived from OpenEmbedded Image Creator (oeic). It includes image that system can boot it in hardware devices.
WIC is supported and below guidelines are shown how to use it to boot Renesas RZ/G devices.

- Enable building WIC image in local.conf (default is enabled) by setting “**WKS_SUPPORT**” to 1:

```
WKS_SUPPORT ?= “1”
```

- Defines additional free disk space created in the image in Kbytes (keep default value if unsure):

```
IMAGE_ROOTFS_EXTRA_SPACE = "1048576"
```

- Select wks file to be built by setting “**WKS_DEFAULT_FILE**” (keep default value if unsure). Currently, there are 2 types of wks defined in “meta-renesas/meta-rz-common/wic” for uSD/eMMC (channel 0) and uSD (channel 1).
- Building your desired core-image by running “bitbake core-image-x”. “core-image-x” should be one of following options:
  - core-image-minimal
  - core-image-bsp
  - core-image-weston
  - core-image-qt

- There are 2 files *wic.bmap and *wic.gz in deploy folder after building successfully. Example:
  - core-image-minimal-smarc-rzg2l.wic.bmap
  - core-image-minimal-smarc-rzg2l.wic.gz

(9) Software bill of materials (SBoM)

Software package data exchange (SPDX) is an open standard for SBoM that identifies and catalogs components, licenses, copyrights, security references, and other metadata relating to software.

SPDX is supported and the guidelines below show how to use it:

- Enable creating SPDX in local.conf (default is disabled) by uncommenting out below line:

```
#INHERIT += "create-spdx"
```

- Select below optional features to be supported for SDPX by enable in local.conf (all is disabled by default):
  - **SPDX_PRETTY**: Make generated files more human readable (newlines, indentation)

```
SPDX_PRETTY = "1"
```

  - **SPDX_ARCHIVE_PACKAGED**: Add compressed archives of the files in the generated target packages in tar.gz files.

```
SPDX_ARCHIVE_PACKAGED = "1"
```

- SPDX is created and deployed in “tmp/deploy/spdx/$MACHINE”. All information can be checked here.s

Note: There is an issue when building SDK (example bitbake core-image-weston -c populate_sdk) with SBoM SPDX support:

```
| ERROR: core-image-weston-1.0-r0 do_populate_sdk: Error executing a python function
 in exec_func_python() autogenerated:
| *** 1078:        return self._accessor.open(self, flags, mode)
|     1079:
|     1080:    def _raw_open(self, flags, mode=0o777):
|     1081:        """
|     1082:        Open the file pointed by this path and return a file descriptor,
|Exception: FileNotFoundError:
|[Errno 2] No such file or directory: 'tmp/work/smarc_rzg2l-poky-linux/core-image-we
ston/1.0-r0/spdx/sdk-work/poky-glibc-x86_64-core-image-weston-aarch64-smarc-rzg2l-ta
rget.spdx.json'
```

To fix this, please apply below change in "**poky/meta/classes/populate_sdk_base.bbclass**":

```
-do_populate_sdk[cleandirs] = "${SDKDEPLOYDIR}"
+do_populate_sdk[cleandirs] += "${SDKDEPLOYDIR}"
```

(10) Qt all demonstrations to build with core-image-qt (for RZ/G2L and RZ/G2LC)

If you include all QT5 Demos in the core-image-qt. Before you build, run the following commands:

```
$ rm -r meta-qt5
$ git clone https://github.com/meta-qt5/meta-qt5.git
$ cd meta-qt5
$ git checkout -b tmp c1b0c9f546289b1592d7a895640de103723a0305
$ git cherry-pick 77b6060cef9337b184100083746c2e35f531be74
$ cd ..
```

And please enable QT_DEMO in the "`~/rzg_vlp_${PACKAGE_VERSION}/build/conf/local.conf`" as below:

```
QT_DEMO = "1"
```

(11) Real time performance

If you want to use the kernel which improves the real-time performance, please add the line below to the `local.conf`.

```
IS_RT_BSP="1"
```

3. Preparing the SD Card

You can prepare the microSD card by the following 2 methods. Please select one of them and follow the steps.

- 3.1 Write files to the microSD card (used wic image)
- 3.2 Write files to the microSD card (not used wic image)

3.1 Write files to the microSD card (used wic image)

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 4**.

Table 6. File and directory in the micro SD card

Type/Number	Filesystem	Contents
Primary #1	FAT32	Flash_Writer_SCIF_<device, memory size>.mot bl2_bp_smarc-<device>.srec bl2_bp_esd-smarc-<device>_pmic.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

Note *1) Please refer to 2.3(8) WIC image for partition size specifications.

3.2 Write files to the microSD card (not used wic image)

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

Please format the card according to the following steps before using the card:

(1) Non-connect microSD card to Linux Host PC

```
$ lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda      8:0    0  30.9G  0 disk
 ├─sda1   8:1    0   512M  0 part /boot/efi
 ├─sda2   8:2    0     1K  0 part
 └─sda5   8:5    0  30.3G  0 part /
sr0     11:0    1  1024M  0 rom
```

(2) Connect microSD card to Linux Host PC with USB adapter

(3) Check the device name which is associated to the microSD card.

```
$ lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda      8:0    0  30.9G  0 disk
 ├─sda1   8:1    0   512M  0 part /boot/efi
 ├─sda2   8:2    0     1K  0 part
```

```
 └─sda5   8:5    0  30.3G  0 part /
sdb      8:16   1  29.7G  0 disk
 └─sdb1   8:17   1  29.7G  0 part
sr0     11:0    1  1024M  0 rom
```

The message above shows the card associated with the `/dev/sdb`. Be careful not to use the other device names in the following steps.

(4) Unmount automatically mounted microSD card partitions

If necessary, unmount all mounted microSD card partitions.

```
$ df
Filesystem     1K-blocks     Used Available Use% Mounted on
udev              745652        0    745652   0% /dev
   :
   snip
   :
/dev/sdb1         511720     4904    506816   1% /media/user/A8D3-393B
$ sudo umount /media/user/A8D3-393B
```

If more than one partition has already been created on microSD card, unmount all partitions.

(5) Change the partition table

microSD card needs two partitions as listed in the following table.

Table 7. Partitions of microSD card

Type/Number	Size	Filesystem	Contents
Primary #1	500MB	FAT32	
Primary #2	All remaining	Ext4	root filesystem Linux kernel Device tree

Set the partition table using the fdisk command like this.

```
$ sudo fdisk /dev/sdb
Welcome to fdisk (util-linux 2.34).
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 0x6b6aac6e.

Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): (Push the enter key)
First sector (2048-62333951, default 2048): (Push the enter key)
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-62333951, default 62333951): +500M

Created a new partition 1 of type 'Linux' and of size 500 MiB.
Partition #1 contains a vfat signature.

Do you want to remove the signature? [Y]es/[N]o: Y

The signature will be removed by a write command.

Command (m for help): n
Partition type
   p   primary (1 primary, 0 extended, 3 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (2-4, default 2): (Push the enter key)
First sector (1026048-62333951, default 1026048): (Push the enter key)
Last sector, +/-sectors or +/-size{K,M,G,T,P} (1026048-62333951, default 62333951): (P
ush the enter key)

Created a new partition 2 of type 'Linux' and of size 29.2 GiB.

Command (m for help): p
Disk /dev/sdb: 29.74 GiB, 31914983424 bytes, 62333952 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: 0x6b6aac6e

Device     Boot   Start      End  Sectors  Size Id Type
/dev/sdb1          2048  1026047  1024000  500M 83 Linux
/dev/sdb2       1026048 62333951 61307904 29.2G 83 Linux

Filesystem/RAID signature on partition 1 will be wiped.

Command (m for help): t
Partition number (1,2, default 2): 1
Hex code (type L to list all codes): b

Changed type of partition 'Linux' to 'W95 FAT32'.

Command (m for help): w
The partition table has been altered.
Syncing disks.
```

Then, check the partition table with the commands below:

```
$ partprobe
$ sudo fdisk -l /dev/sdb
Disk /dev/sdb: 29.74 GiB, 31914983424 bytes, 62333952 sectors
Disk model: Maker name etc.
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: 0x6b6aac6e

Device     Boot   Start      End  Sectors  Size Id Type
/dev/sdb1          2048  1026047  1024000  500M  b W95 FAT32
/dev/sdb2       1026048 62333951 61307904 29.2G 83 Linux
```

(6) Format and mount the partitions

If the partitions were automatically mounted after step (4), please unmount them according to step (3).

Then format the partitions using the command below:

```
$ sudo mkfs.vfat -v -c -F 32 /dev/sdb1
mkfs.fat 4.1 (2017-01-24)
/dev/sdb1 has 64 heads and 32 sectors per track,
hidden sectors 0x0800;
logical sector size is 512,
using 0xf8 media descriptor, with 1024000 sectors;
drive number 0x80;
filesystem has 2 32-bit FATs and 8 sectors per cluster.
FAT size is 1000 sectors, and provides 127746 clusters.
There are 32 reserved sectors.
Volume ID is a299e6a6, no volume label.
Searching for bad blocks 16848... 34256... 51152... 68304... 85072... 102096... 11937
6... 136528... 153552... 170576... 187472... 204624... 221648... 238928... 256208... 2
73744... 290768... 308048... 325328... 342480... 359504... 376656... 393680... 41057
6... 427216... 444624... 462032... 479184... 495952...

$ sudo mkfs.ext4 -L rootfs /dev/sdb2
```

```
mke2fs 1.45.5 (07-Jan-2020)
Creating filesystem with 7663488 4k blocks and 1916928 inodes
Filesystem UUID: 63dddb3f-e268-4554-af51-1c6e1928d76c
Superblock backups stored on blocks:
  32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
  4096000

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done
```

(7) Remount microSD card

After format, remove the card reader and connect it again to mount the partitions.

(8) Write files to the microSD card

Check the mount point name with df command.

```
$ df
Filesystem     1K-blocks     Used Available Use% Mounted on
udev              745652        0    745652   0% /dev
   :
   snip
   :
/dev/sdb1         510984       16    510968   1% /media/user/A299-E6A6
/dev/sdb2       30041556    45080  28447396   1% /media/user/rootfs
```

Expand rootfs to the second partition.

```
$ cd /media/user/rootfs
$ sudo tar jxvf $WORK/build/tmp/deploy/images/<board>/<root filesystem>
```

Please replace *<board>* are listed in the **Table 2**.

Table 8. File and directory in the microSD card

Type/Number	Size	Filesystem	Contents
Primary #1	Size specified when the partition was created.	FAT32	
Primary #2	Size specified when the partition was created.	Ext4	./ ├── bin ├── boot │ ├── Image-<board>.bin │ └── Image-<device>-smarc.dtb ├── dev ├── etc ├── home ├── lib ├── media ├── mnt ├── proc ├── run ├── sbin ├── 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).

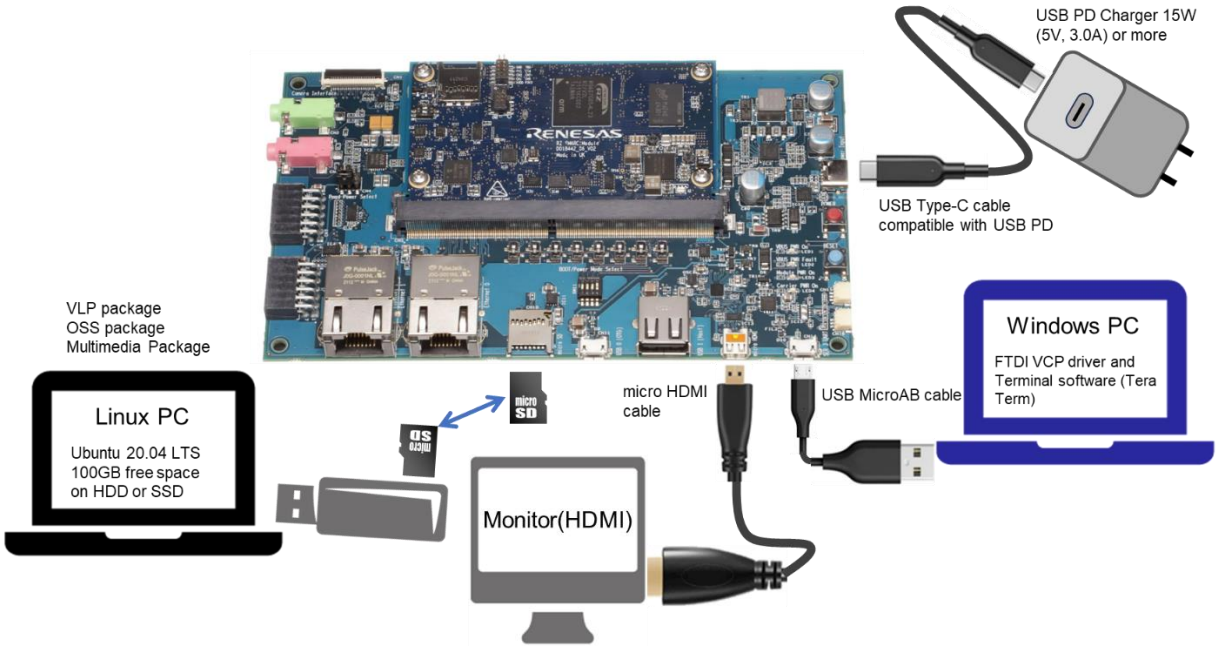


Figure 2. Operating environment

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.

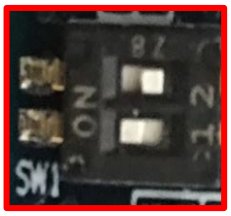
SW11-4	Input voltage selection
OFF	Input 9V
ON	Input 5V

4.1.2 How to set SW1

Please set the SW1 settings to follows for RZ/G2L 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).

- 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.



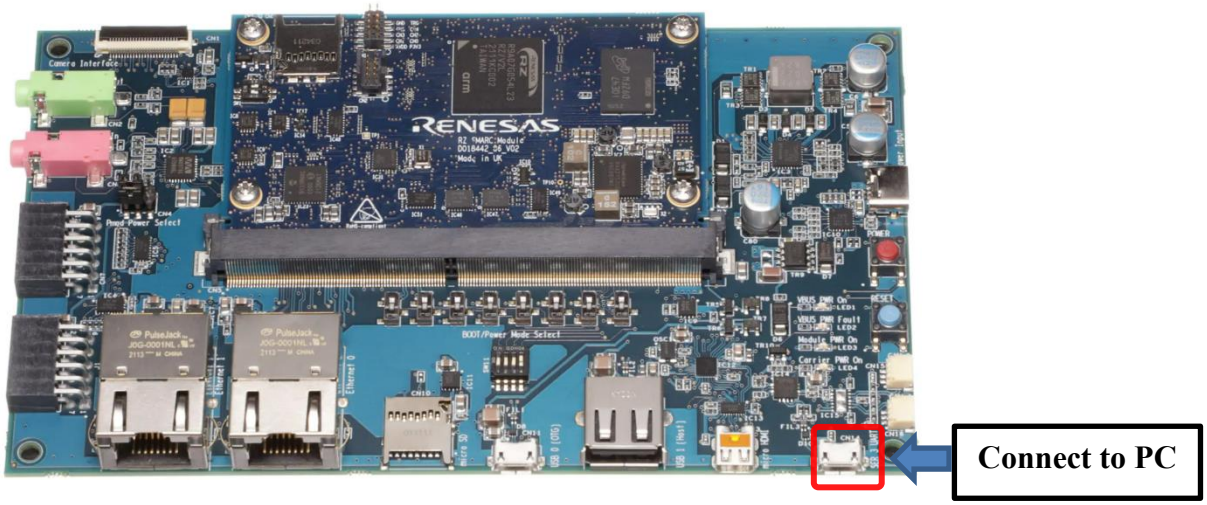
SW1-2	OFF
SW1-1	ON

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 the 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 describing 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.

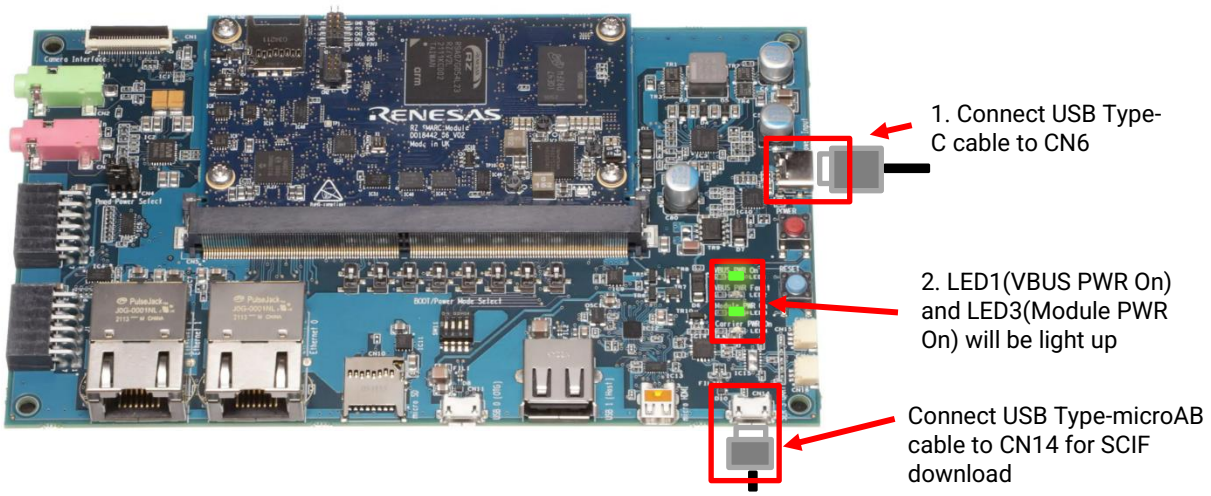


Figure 4. Connecting Power Supply

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.

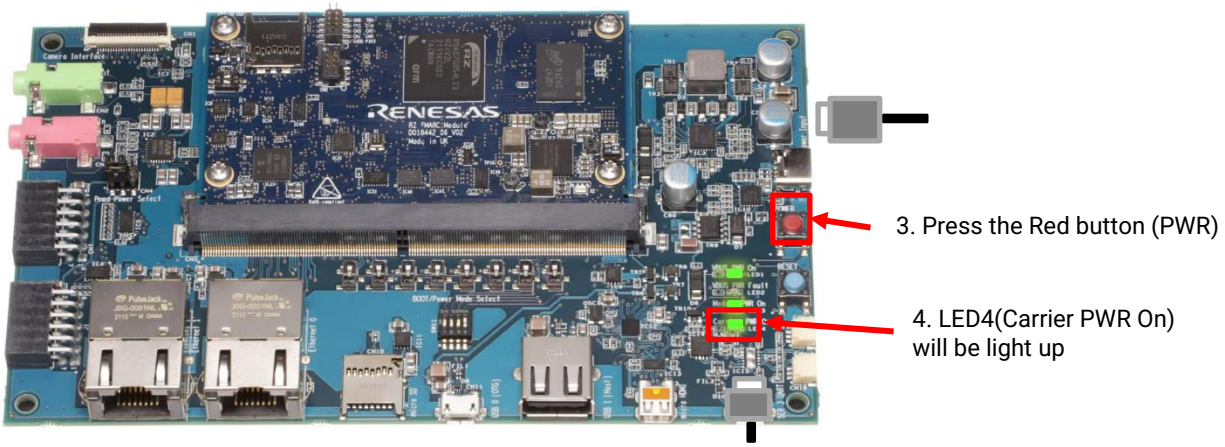


Figure 5. Power ON

4.2.2 Building files to write

The evaluation boards use the files in the Table 9 as boot loaders. The boot loaders files are generated by the build instruction in section 2.2. Please refer to **Table 4**. Once builded copy these files to a PC which runs serial terminal software.

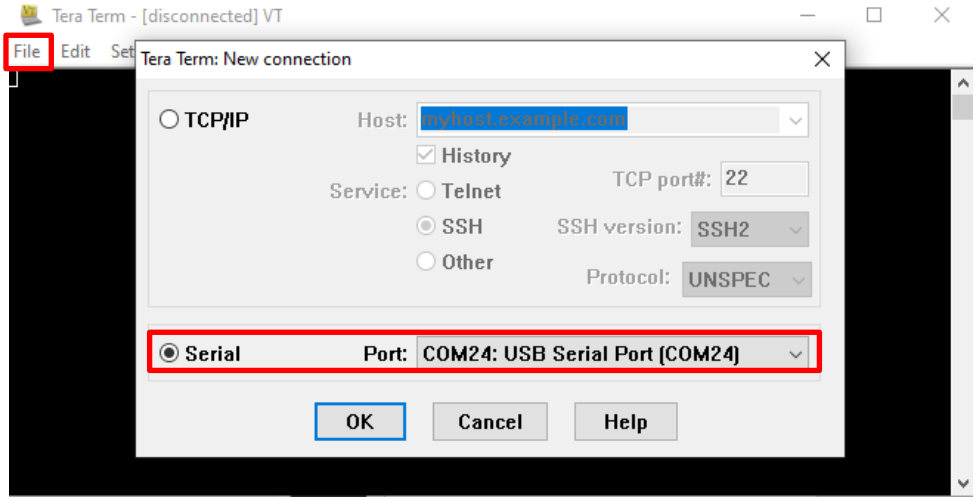
Table 9. File names of Boot loader

Board	File name of Boot loader
RZ/G2L Evaluation Board Kit PMIC version	• bl2_bp-smarc-rzg2l_pmic.srec • fip-smarc-rzg2l_pmic.srec
RZ/G2LC Evaluation Board Kit	• bl2_bp-smarc-rzg2lc.srec • fip-smarc-rzg2lc.srec
RZ/G2UL Evaluation Board Kit	• bl2_bp-smarc-rzg2ul.srec • fip-smarc-rzg2ul.srec

4.2.3 Settings

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

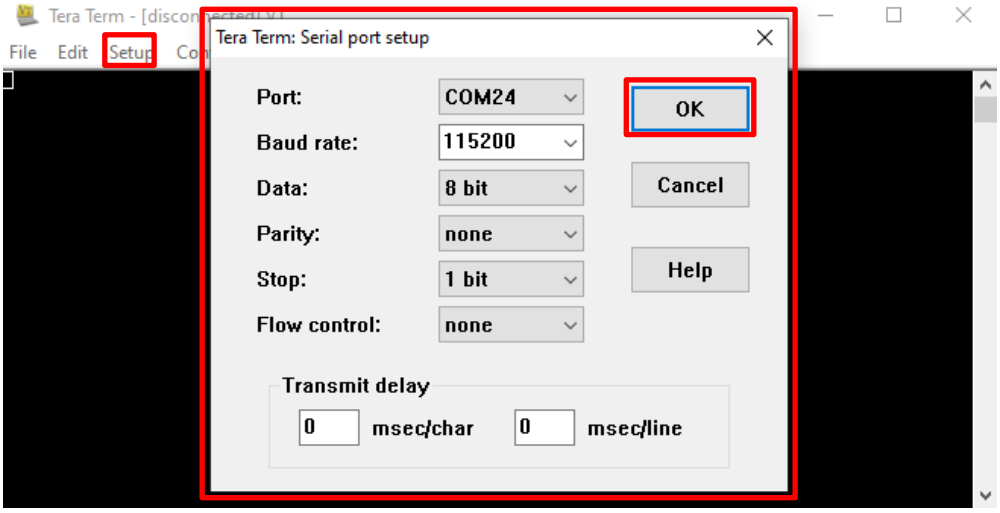
1. Bring up the terminal software and select the “File” > “New Connection” to set the connection on the software.



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

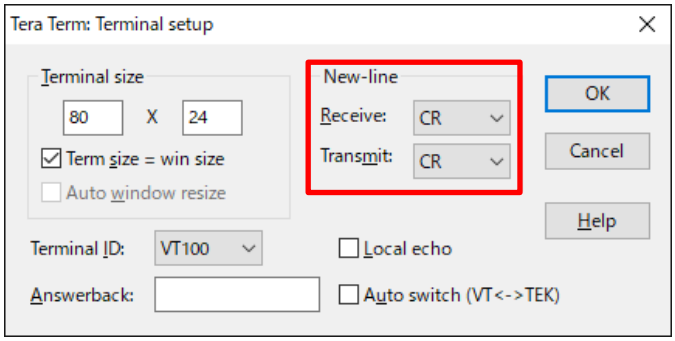
Set the settings about serial communication protocol on a terminal software as below:

- 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 10. 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.

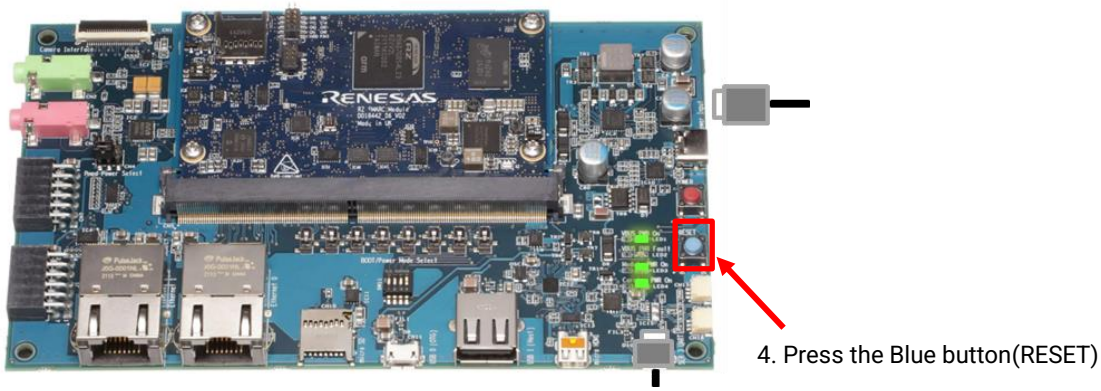
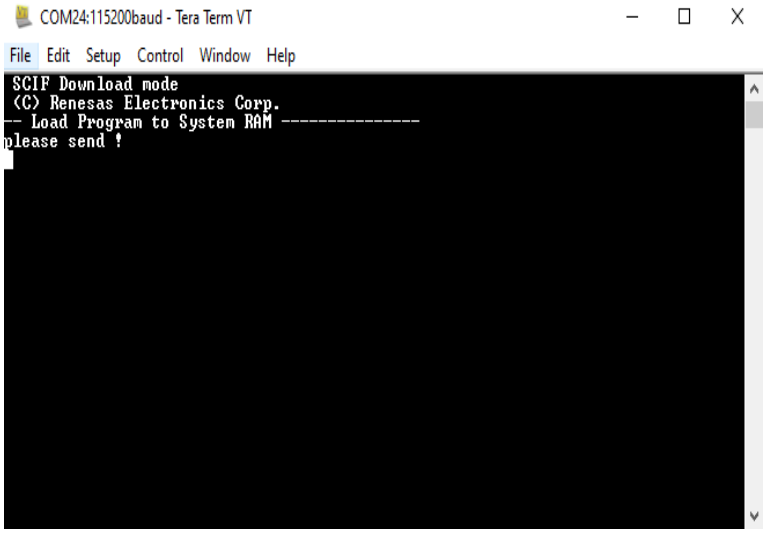


Figure 6. Press the RESET button



4.3 Download Flash Writer to RAM

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 !
```

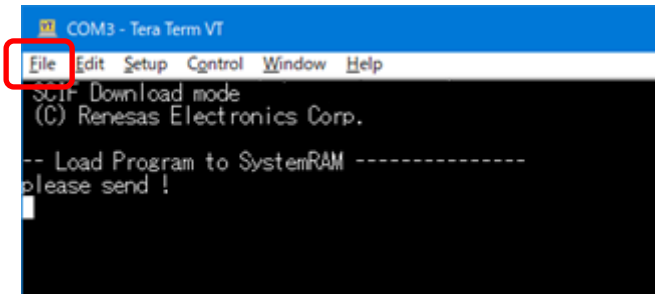
Send an image of Flash Writer using terminal software after the message “please send !” is shown. Please refer to the Table 11 below to know which file name of Flash Writer should be sent.

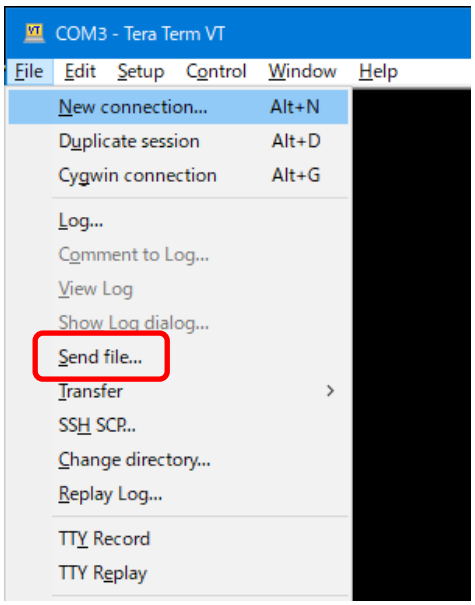
Table 11. File names of Flash Writer

Board name	File name of Flash Writer
RZ/G2L Evaluation Board Kit PMIC version	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

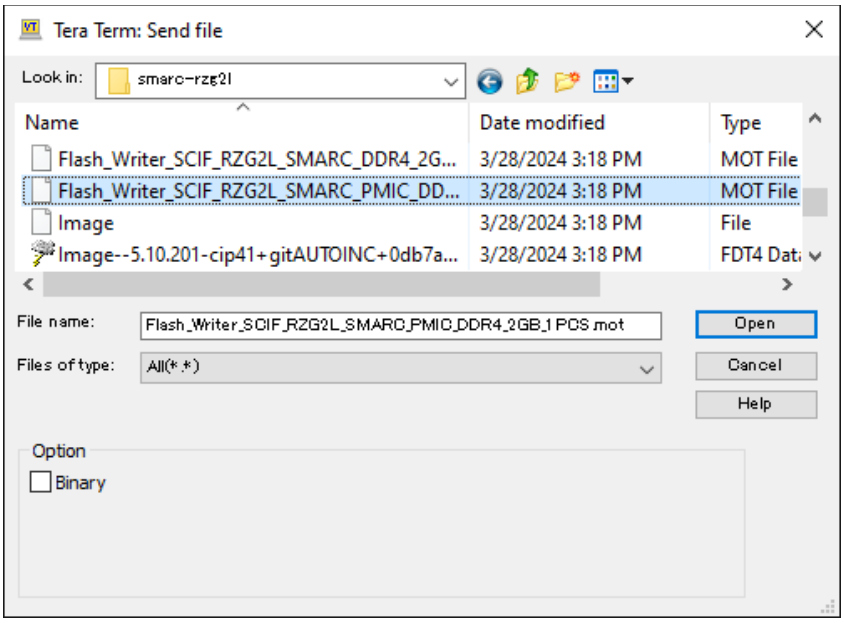
Below is a sample procedure with Tera Term.

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

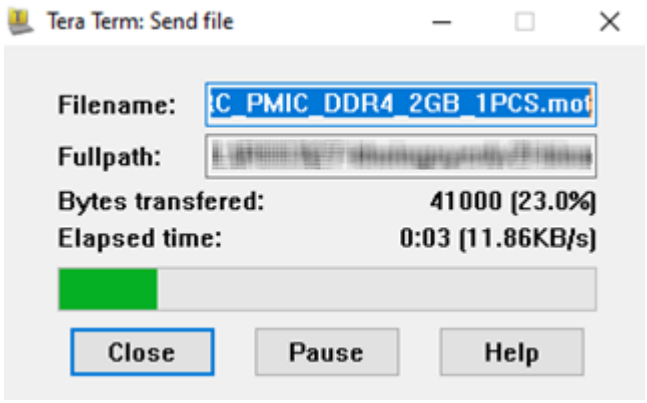




Then, select the image to be send and click “Open” button.



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 8**.

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 PMIC version.:

```
>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-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'1D200
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'0001D200
 SpiFlashMemory End Address  : H'000CC73F
============================================================
```

```
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 12. Address for sending each loader binary file

RZ/G2L Evaluation Board Kit PMIC version

File name	Address to load to RAM	Address to save to ROM
bl2_bp-smarc-rzg2l_pmic.srec	11E00	00000
fip-smarc-rzg2l_pmic.srec	00000	1D200

RZ/G2LC Evaluation Board Kit

File name	Address to load to RAM	Address to save to ROM
bl2_bp-smarc-rzg2lc.srec	11E00	00000
fip-smarc-rzg2lc.srec	00000	1D200

RZ/G2UL Evaluation Board Kit

File name	Address to load to RAM	Address to save to ROM
bl2_bp-smarc-rzg2ul.srec	11E00	00000
fip-smarc-rzg2ul.srec	00000	1D200

4.5 Change Back to Normal Boot Mode

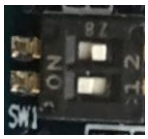
To set the board to SPI Boot mode, set the SW11 as below:

Table 13. SW11

1	2	3	4
OFF	OFF	OFF	ON

Note:-

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 power of the board by pressing the reset button SW10.

```
U-Boot 2021.10 (Mar 31 2022 - 03:57:20 +0000)

CPU:   Renesas Electronics K rev 16.10
Model: smarc-rzg2l
DRAM:  1.9 GiB
MMC:   sd@11c00000: 0, sd@11c10000: 1
Loading Environment from MMC... OK
In:    serial@1004b800
Out:   serial@1004b800
Err:   serial@1004b800
Net:
Error: ethernet@11c20000 address not set.
No ethernet found.

Hit any key to stop autoboot:  2  1  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.

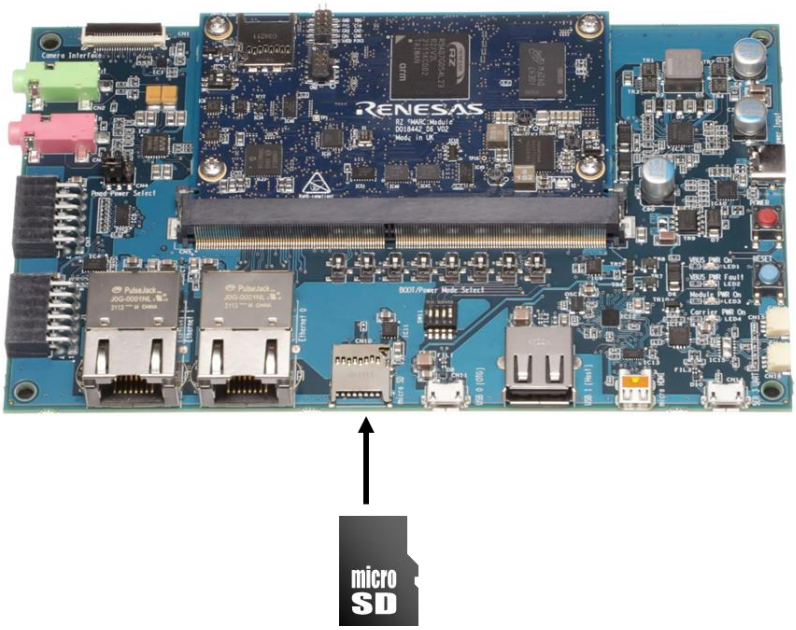


Figure 7. Set micro SD card to SMARC-EVK

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.

```
U-Boot 2021.10 (Mar 31 2022 - 03:57:20 +0000)

CPU:   Renesas Electronics K rev 16.10
Model: smarc-rzg2l
DRAM:  1.9 GiB
MMC:   sd@11c00000: 0, sd@11c10000: 1
Loading Environment from MMC... OK
In:    serial@1004b800
Out:   serial@1004b800
Err:   serial@1004b800
Net:
Error: ethernet@11c20000 address not set.
No ethernet found.

Hit any key to stop autoboot:  2  1  0
switch to partitions #0, OK
mmc1 is current device
19857920 bytes read in 1229 ms (15.4 MiB/s)
39079 bytes read in 6 ms (6.2 MiB/s)
Moving Image from 0x48080000 to 0x48200000, end=49560000
```

```
## Flattened Device Tree blob at 48000000
   Booting using the fdt blob at 0x48000000
   Loading Device Tree to 0000000057ff3000, end 0000000057fff8a6 ... OK

Starting kernel ...

[    0.000000] Booting Linux on physical CPU 0x0000000000 [0x412fd050]
[    0.000000] Linux version 5.10.83-cip1-yocto-standard (oe-user@oe-host) (aarc
 :
 :
Poky (Yocto Project Reference Distro) 3.1.14 smarc-rzg2l ttySC0

BSP: RZG2L/RZG2L-SMARC-EVK/3.0.3
LSI: RZG2L
Version: 3.0.3
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 steps (1) – (6) 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.

For building general applications:

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

For building Qt applications:

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

The resulting SDK installer will be located 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 poky-glibc-x86_64-core-image-weston-aarch64-<board>-toolchain-3.1.33.sh
```

Or

```
$ sudo sh poky-glibc-x86_64-core-image-qt-aarch64-<board>-toolchain-3.1.33.sh
```

RZ/G2L Evaluation Board Kit PMIC version:	smarc-rzg2l
RZ/G2LC Evaluation Board Kit:	smarc-rzg2lc
RZ/G2UL Evaluation Board Kit:	smarc-rzg2ul

Note) The SDK build may fail depending on the build environment. At that time, please run the building again after a period. Or build it again from scratch with the commands below.

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

For building general applications:

```
$ MACHINE=<board> bitbake core-image-weston -c populate_sdk
```

For building Qt applications:

```
$ MACHINE=<board> bitbake core-image-qt -c populate_sdk
```

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 ./poky-glibc-x86_64-core-image-weston-sdk-aarch64-toolchain-<version>.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 ./rzg_vlp_${PACKAGE_VERSION}/build/tmp/deploy/
sdk/poky-glibc-x86_64-core-image-weston-aarch64-smarc-rzg2l-toolchain-x.x.xx.sh
Poky (Yocto Project Reference Distro) SDK installer version x.x.xx
==========================================================
Enter target directory for SDK (default: /opt/poky/x.x.xx):
You are about to install the SDK to "/opt/poky/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/poky/x.x.xx/environment-setup-aarch64-poky-linux
 $ . /opt/poky/x.x.xx/environment-setup-armv7vet2hf-neon-vfpv4-pokymllib32-linux-gnuea
bi
```

Step 2. Set up cross-compile environment:

```
$ source /<Location in which SDK is extracted>/environment-setup-aarch64-poky-linux
```

Note:

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

```
$ source /opt/poky/x.x.xx/environment-setup-aarch64-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 “hello”) 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. 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.

```
BSP: RZG2L/RZG2L-SMARC-EVK/${PACKAGE_VERSION}
LSI: RZG2L
Version: ${PACKAGE_VERSION}
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, 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 set up 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 4168466783f06fa7f2aa5782c597803a6882ed2f
```

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 14**.

Table 14. BOARD option

Target board	BOARD option <*board*>	Image to be generated
smarc-rzg2l-pmic	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)

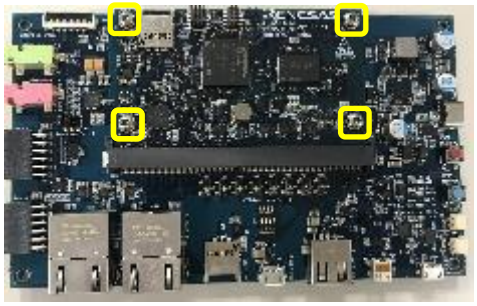
8.3 How to replace the SMARC Module Board

Please be careful when replacing the board as follows.

1. Remove the four screws.

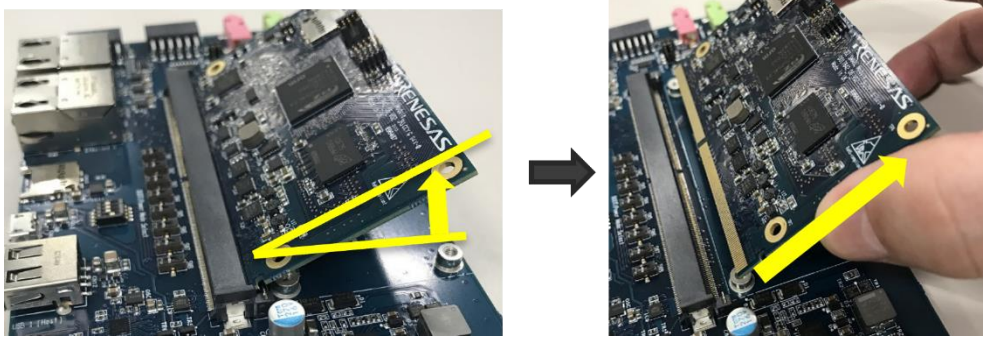
Note: The screw thread is a special shape, so be careful not to crush the screw thread.

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

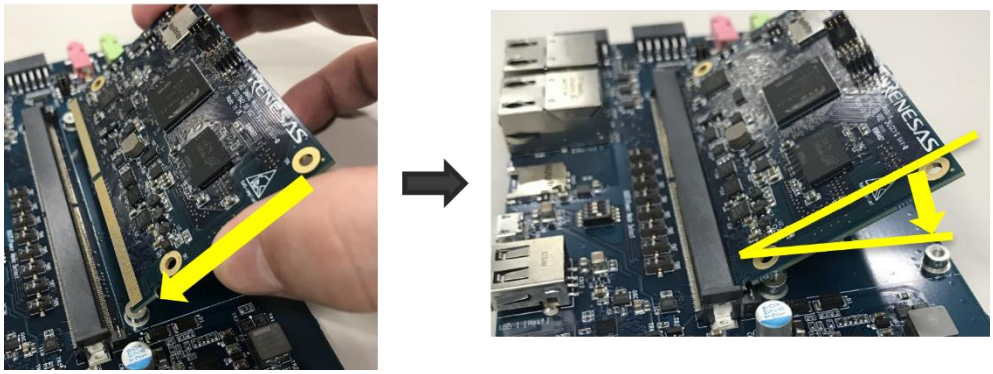


Specially shaped screw threads

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

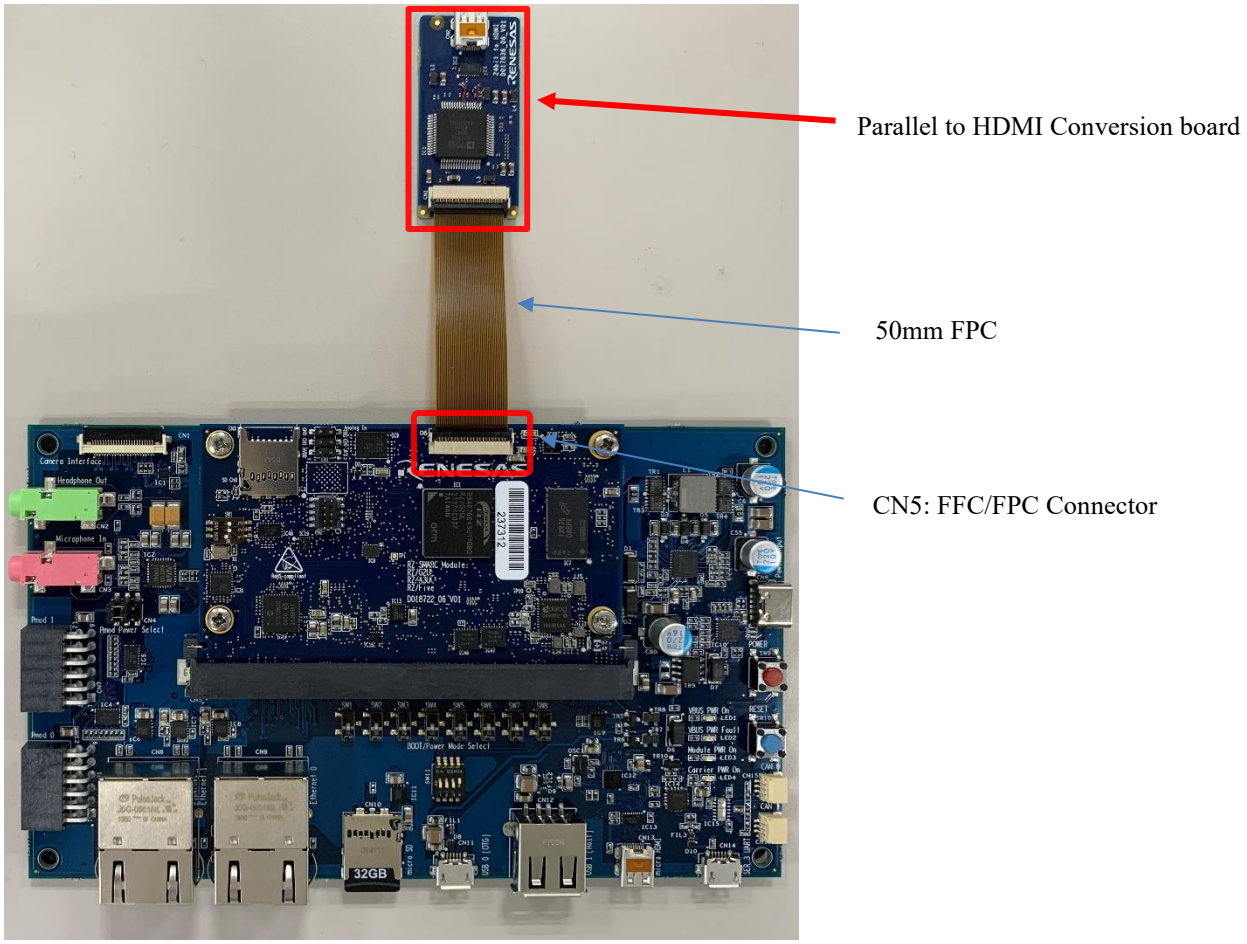


3. Insert the replacement the board diagonally, then roll the SMARC board parallel to the board and fix it with screws.



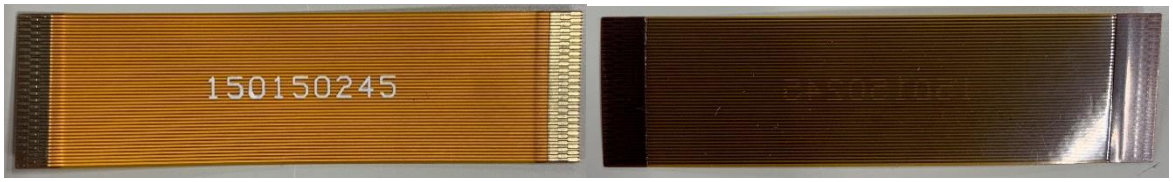
8.4 How to connect Parallel to HDMI Conversion board for RZ/G2UL Evaluation kit (smarc-rzg2ul)

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



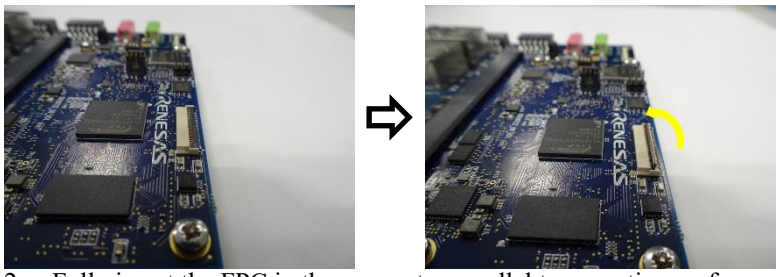
The termination procedure of the 50mm FPC is follows.

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

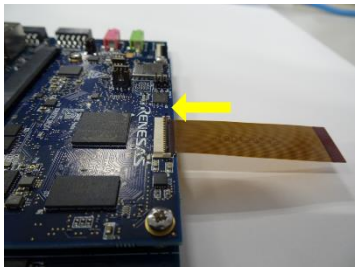


Top surface Bottom surface

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

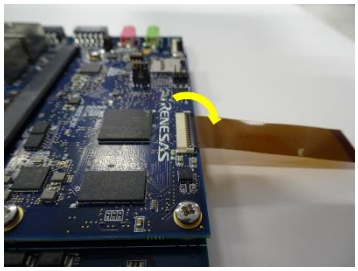


2. Fully insert the FPC in the connector parallel to mounting surface, with the exposed conductive traces facing down



3. Rotate down the actuator until firmly closed.

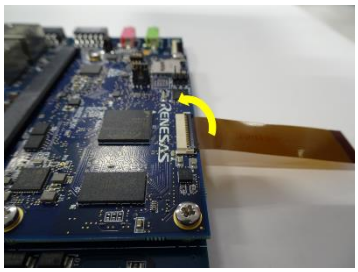
Note: The PFC must be fully inserted in the connector. If not fully inserted, the actuator will not close property. Should this be the case, lift up the actuator and repeat the process (starting with Step 1 above)



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 on RZ/G2L, RZ/G2LC, RZ/G2UL, and RZ/G2L are described.

8.5.1 Rebuild rootfs

Build a rootfs according to Release Note. After that, please rebuild the rootfs with the command below.

```
$ cd ${your build directory} (ex.$ cd ~/rzg_vlp_v3.0.7)
$ 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 15**, and corresponding bootloader files and specified address information are depending on each target board as described in **Table 16**.

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 PMIC version.:

```
>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-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 15. Address of EXT_CSD register of eMMC for eMMC boot

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

Table 16. Address for sending each loader binary file for eMMC boot

RZ/G2L Evaluation Board Kit PMIC version

File name	Partition to save to eMMC	Address to save to eMMC	Address to load to RAM
bl2_bp-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

RZ/G2UL Evaluation Board Kit

File name	Partition to save to eMMC	Address to save to eMMC	Address to load to RAM
bl2_bp-smarc-rzg2ul.srec	1	00000001	11E00
fip-smarc-rzg2ul.srec	1	00000100	00000

8.5.3 Create a microSD card to boot linux for eMMC boot

Create a microSD card by Section 3. After that, please go back to following instructions before un-mounting a microSD card.

Copy a kernel image, a 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 Setting U-boot and writing rootfs to eMMC

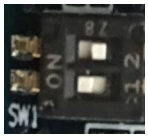
To set the board to eMMC Boot mode, set the SW11 as below:

Table 17. 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 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).

Refer to Section 4.5 and set the u-boot environment variables (please ignore switch settings in that section). Then, turn off and on the power of the board by pressing the reset button SW10.

After booting Linux, please login as root and create partitions on eMMC.

```
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 eMMC and write the kernel, the device tree, and the 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.

```
U-Boot 2021.10 (Apr 22 2022 - 03:04:59 +0000)

CPU:   Renesas Electronics K rev 16.15
Model: smarc-rzg2l
DRAM:  1.9 GiB
MMC:   sd@11c00000: 0, sd@11c10000: 1
Loading Environment from MMC... OK
In:    serial@1004b800
Out:   serial@1004b800
Err:   serial@1004b800
Net:   eth0: ethernet@11c20000
Hit any key to stop autoboot:  0
=>
```

Set environment variables to boot from eMMC. The following variables are for the RZ/G2L board. Please replace the file names in “bootcmd” according to the Release Note when you use other boards and set 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-r9a07g054l2-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.

RZ/G2L can also boot from a micro-SD card, this is booting mode 0 called "Booting from eSD" in the User Manual.

Note) In this release, the RZ/G2L and RZ/G2LC boards must update u-boot to boot from eSD automatically into the kernel without modifying the device tree. This step is mandatory as per the Release Note.

8.6.1 Prepare micro SD card

Prepare the micro SD card using the wic image file**.**

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

Table 18. File and directory in the micro SD card

Type/Number	Filesystem	Contents
Primary #1	FAT32	Flash_Writer_SCIF_<device, memory size>.mot bl2_bp_smarc-<device>.srec bl2_bp_esd-smarc-<device>_pmic.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 in the SOM module slot.

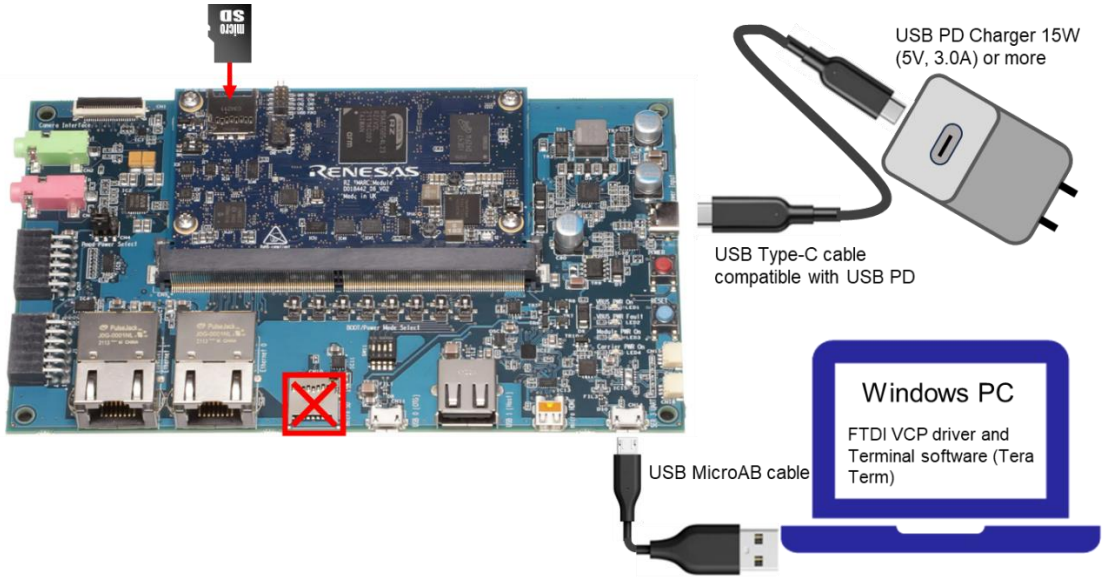


Figure 8. Operating environment for eSD boot

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: v2.9(release):cc18695-dirty
NOTICE:  BL2: Built : 04:12:39, Dec 14 2023
NOTICE:  BL2: SD boot from partition 0
NOTICE:  BL2: Load dst=0x1f840 src=(p:0)0x10000(128) len=0x10(1)
NOTICE:  BL2: SD boot from partition 0
NOTICE:  BL2: Load dst=0x1f9a0 src=(p:0)0x10010(128) len=0x28(1)
NOTICE:  BL2: SD boot from partition 0
NOTICE:  BL2: Load dst=0x44000000 src=(p:0)0x10090(128) len=0x6069(49)
NOTICE:  BL2: SD boot from partition 0
NOTICE:  BL2: Load dst=0x1f840 src=(p:0)0x10000(128) len=0x10(1)
NOTICE:  BL2: SD boot from partition 0
NOTICE:  BL2: Load dst=0x1f9a0 src=(p:0)0x10010(128) len=0x28(1)
NOTICE:  BL2: Load dst=0x1f9a0 src=(p:0)0x10038(128) len=0x28(1)
NOTICE:  BL2: SD boot from partition 0
NOTICE:  BL2: Load dst=0x50000000 src=(p:0)0x16100(176) len=0xbae68(1496)
NOTICE:  BL2: Booting BL31
NOTICE:  BL31: v2.9(release):cc18695-dirty
NOTICE:  BL31: Built : 04:12:39, Dec 14 2023


U-Boot 2021.10 (Dec 15 2023 - 06:47:44 +0000)

CPU:   Renesas Electronics CPU rev 1.0
Model: smarc-rzg2l
DRAM:  1.9 GiB
WDT:   watchdog@0000000012800800
WDT:   Started with servicing (60s timeout)
MMC:   sd@11c00000: 0, sd@11c10000: 1
Loading Environment from MMC... *** Warning - bad CRC, using default environment

In:    serial@1004b800
Out:   serial@1004b800
Err:   serial@1004b800
U-boot WDT started!
Net:
Error: ethernet@11c20000 address not set.
No ethernet found.

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 5.10.184-cip36-yocto-standard (oe-user@oe-host) (aarc
 :
```

```
 :
Poky (Yocto Project Reference Distro) 3.1.33 smarc-rzg2l ttySC0

BSP: RZG2L/RZG2L-SMARC-EVK/3.0.7
LSI: RZG2L
Version: 3.0.7
smarc-rzg2l login: root
root@smarc-rzg2l:~#
```

8.7 Docker

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

(1) Add layers to enable Docker

After completing step (3) in section 2.2, run the following commands. Once executed, proceed until step (6) in section 2.2 to build the VLP with Docker enabled.

```
$ cd ~/rzg_vlp_${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 section 4 and 5, boot the evaluation board, and proceed to the next step.

(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     |
+-----------+--
```

At 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 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-v3.0.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 19. Support device drivers

Device Driver	Documents
Kernel Core	r01us0468ej0xxx-rz-g_Kernel_Core_UME.pdf
Direct Memory Access Controller (DMAC)	r01us0480ej0xxx-rz-g_DMAC_UME.pdf
Multi-function Timer Unit 3a (MTU3a)	r01us0476ej0xxx-rz-g_MTU3a_UME.pdf
Port Output Enable 3 (POE3)	r01us0552ej0xxx-rz-g_POE3_UME.pdf
General PWM Timer (GPT)	r01us0484ej0xxx-rz-g_GPT_UME.pdf
Watchdog Timer (WDT)	r01us0479ej0xxx-rz-g_WDT_UME.pdf
Serial Communication Interface with FIFO A (SCIFA)	r01us0483ej0xxx-rz-g_SCIFA_UME.pdf
Renesas Serial Peripheral Interface (RSPI)	r01us0481ej0xxx-rz-g_SPI_UME.pdf
SPI Multi IO Bus Controller	r01us0482ej0xxx-rz-g_SPI_Multi IO_UME.pdf
I2C Bus Interface	r01us0477ej0xxx-rz-g_I2C_UME.pdf
Serial Sound Interface	r01us0490ej0xxx-rz-g_SSI_UME.pdf
RS-CANFD Interface	r01us0478ej0xxx-rz-g_CANFD_UME.pdf
Gigabit Ethernet Interface	r01us0475ej0xxx_rz-g_Gigabit_Ethernet_UME.pdf
A/D Converter	r01us0487ej0xxx-rz-g_AD_Converter_UME.pdf
USB 2.0 Host	r01us0485ej0xxx-rz-g_USB2.0_Host_UME.pdf
USB 2.0 Function	r01us0491ej0xxx-rz-g_USB2.0_Function_UME.pdf
LCD Controller (LCDC)	r01us0489ej0xxx-rz-g_LCDC_UME.pdf
Camera Receiving Unit (CRU)	r01us0499ej0xxx-rz-g_CRU_UME.pdf
SD/MMC Host Interface	r01us0474ej0xxx-rz-g_SD_MMC_UME.pdf
GPIO	r01us0488ej0xxx-rz-g_GPIO_UME.pdf
Power Management	r01us0605ej0xxx-rz-g_Power_Management_UME.pdf
Thermal Sensor Unit (TSU)	r01us0486ej0xxx-rz-g_Thermal_Sensor__UME.pdf
Error Correction Code (ECC)	r01us0647ej0xxx-rz-mpu_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	Sep. 30, 2023	-	First edition issued.
1.01	Jan. 29, 2024	6	Modify the explanation of the patch and update some texts.
1.02	Apr. 24, 2024	6	Add “2.1 Required Host OS” section.
		55	Add “8.6 How to boot from eSD” section.
		59	Add “8.7 Docker” section.
1.03	May 31, 2024	-	VLP/G v3.0.6-update1
		60	Add “8.8 Booting Setup with Ubuntu PC” section.
1.04	Jul. 31, 2024	21	Update “3.2 Write files to the microSD card (not used wic image)” section.
		54	Update “8.6 How to boot from eSD” section.
1.05	Jan. 31, 2025	25	Add link address of SW1 for RZ/G2LC and RZ/G2UL
		40	Display “<-tab->” code in the Makefile
1.06	Mar. 31, 2025	-	Update the explanation for VLP/G v3.0.7.
		15	Added note about Section (12) *Kernel Setting for eSD* in Section 2.3 *Notes*.
1.0.7	Jul. 31, 2025	-	VLP/G v3.0.7-update2 Removed note for Section (12) "Kernel Setting for eSD" from Section 2.3 Notes.

Website and Support

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

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