# SMARC EVK of RZ/G3S

## Linux Start-up Guide

R01US0816EJ0101
Rev.1.01
Sep. 30, 2026

### Introduction

This document provides a guide to prepare RZ/G3S 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/G3S reference board

- RZ/G3S Evaluation board Kit (P/N: RTK9845S33S01000BE) (smarc-rzg3s)
    - RZ/G3S SMARC Module Board (P/N: RTK9845S33C01000BE)
    - RZ SMARC Series Carrier Board (P/N: RTKSMCBB2B01000BE)

### Target Software

- RZ/G3S Verified Linux Package version 4.0.1 or later. (hereinafter referred to as “VLP”)

## Contents

1. Environmental Requirement ..... 4
2. Build Instructions ..... 6
2.1 Required Host OS ..... 6
2.2 Building images ..... 6
2.3 Notes ..... 9
3. Preparing the SD Card ..... 11
4. Reference Board Setting ..... 12
4.1 Preparation of Hardware and Software ..... 12
4.1.1 How to set boot mode and input voltage ..... 13
4.1.2 How to set “VBUS_SEL” rotary switch ..... 15
4.1.3 How to Set “SW_CONFIG” DIP switch ..... 16
4.1.4 How to set “SW_OPT_MUX” DIP switch ..... 16
4.1.5 How to use debug serial (console output) ..... 17
4.2 Startup Procedure ..... 18
4.2.1 Power supply ..... 18
4.2.2 Building files to write ..... 20
4.2.3 Settings ..... 20
4.3 Download Flash Writer to RAM ..... 23
4.4 Write the Bootloader ..... 25
4.5 Change Back to Normal Boot Mode ..... 27
5. Booting and Running Linux ..... 28
5.1 Power on the board and Startup Linux ..... 28
5.2 Shutdown the Board ..... 30
6. Building the SDK ..... 31
7. Application Building and Running ..... 32
7.1 Make an application ..... 32
7.1.1 How to extract SDK ..... 32
7.1.2 How to build Linux application ..... 32
7.2 Run a sample application ..... 34
8. Appendix ..... 35
8.1 How to replace the SMARC Module Board ..... 35
8.2 How to boot from eMMC ..... 36
8.2.1 Writing Bootloader for eMMC Boot ..... 36
8.2.2 Create a microSD card to boot linux for eMMC boot ..... 37
8.2.3 Setting U-boot and writing rootfs to eMMC ..... 38
8.2.4 Setting U-boot for eMMC boot ..... 40

8.3 How to boot from eSD ..... 41
8.3.1 Prepare micro SD card ..... 42
8.3.2 Set SMARC EVK board for eSD boot ..... 42
8.3.3 Power on and boot ..... 44
8.4 Docker ..... 45
8.5 Booting Setup with Ubuntu PC ..... 46
8.6 Device drivers ..... 48

9. Revision History ..... 49

Website and Support ..... 50

# 1. Environmental Requirement

The environment for building the Board Support 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 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.

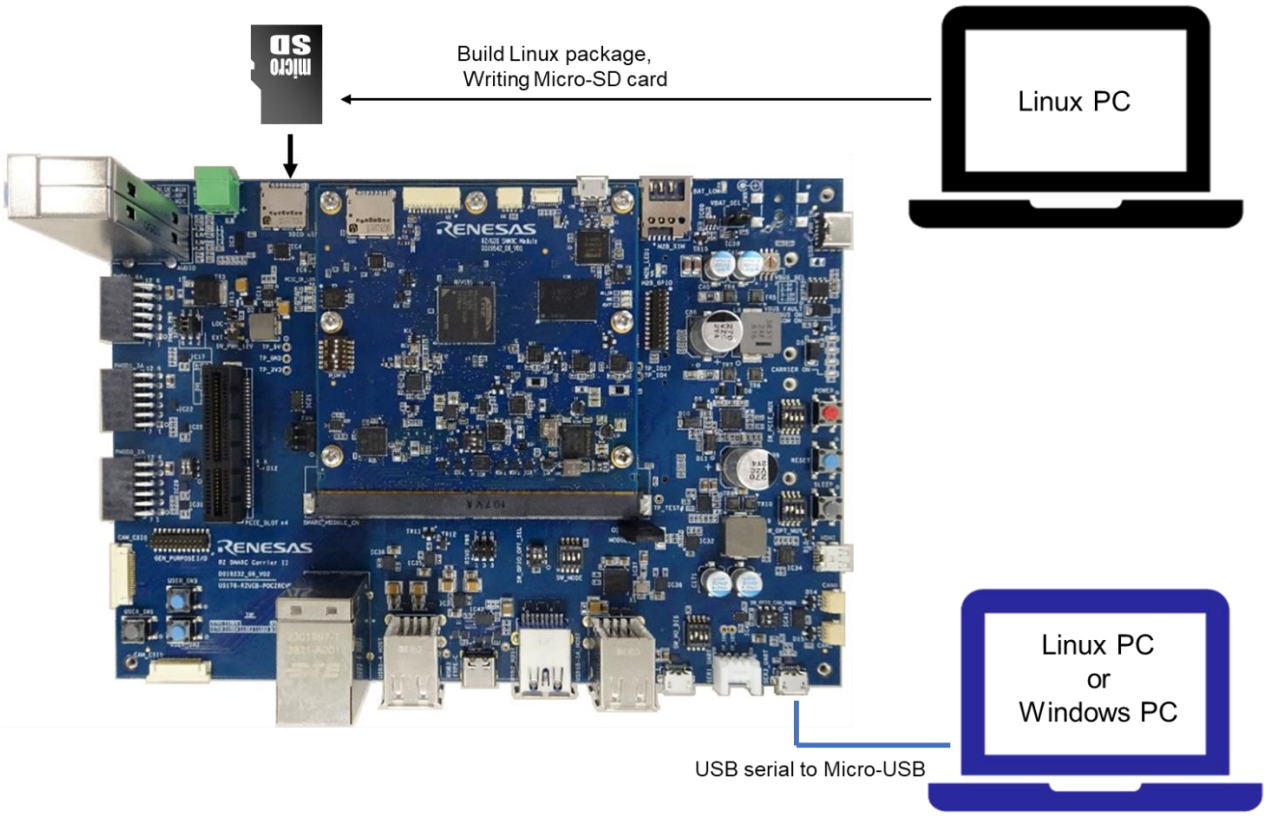



**Figure 1. Recommend environment.**

**Table 1. Equipment and Software for Developing Environments of RZ/G3S Linux Platform**

| Equipment | | Description |
|---|---|---|
| Linux Host PC | | Used as build/debug environment<br>200GB free space on HDD or SSD is necessary |
| | OS | Ubuntu 22.04 LTS<br>64 bit OS must be used.<br>22.04 inside a docker container also OK. |
| 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<br>Tera Term (latest version) is recommended<br>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:<br>• 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. |

# 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 Board Support Package.
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/5.0.14/brief-yoctoprojectqs/

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) 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_2
```

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-rz-vlp-v${PACKAGE_VERSION}.zip
$ tar zxvf ./RTK0EF0045Z0035AZJ-rz-vlp-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 large. Especially in case you are using a Virtual Machine, please check how much disk space you have allocated for your virtual environment.

### (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-rz-distro/conf/templates/vlp-v4-conf/ source \
poky/oe-init-build-env build
```

### (4) Add layers (Optional)

- **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
```

### (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_rz_v${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/G3S 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 “~/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_2/build/own-mirror"
```

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

**(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.

Note) In this version, a fetch error will occur during the build process unless you follow the procedure outlined in the "Restrictions" section of the Release Note. Failure to comply may result in building failure. Please make sure to perform the steps before starting the building process.

Build the target file system image using bitbake

```
$ MACHINE=smarc-rzg3s bitbake core-image-minimal
```

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

**Table 2. Image files for RZ/G3S**

| | | |
|---|---|---|
| **RZ/G3S** | **Linux kernel** | Image |
| | **Device tree file** | r9a08g045s33-smarc.dtb |
| | **root filesystem** | core-image-minimal-smarc-rzg3s.rootfs.tar.bz2 |
| | **Boot loader** | • bl2_bp_spi-smarc-rzg3s.srec<br>• bl2_bp_mmc-smarc-rzg3s.srec<br>• bl2_bp_esd-smarc-rzg3s.srec<br>• bl2_bp_esd-smarc-rzg3s.bin<br>• fip-smarc-rzg3s.srec<br>• fip-smarc-rzg3s.bin |
| | **Flash Writer** | FlashWriter-smarc-rzg3s.mot |
| | **SD image (wic)** | core-image-minimal-smarc-rzg3s.rootfs.wic.gz<br>core-image-minimal-smarc-rzg3s.rootfs.wic.bmap |
| | **Software Bill of Materials (SBOM)** | core-image-minimal-smarc-rzg3s.rootfs.spdx.tar.zst |

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

## 2.3 Notes

### (1) 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 e08a700bfd6160e35d556d2e7a322c7b659b112f
$ cd ..
```

Then run the command below 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/vlp-v4-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.

### (2) Real time performance

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

```
# Un-comment following lines to select RTLinux.
PREFERRED_PROVIDER_virtual/kernel = "linux-renesas-rt"
PREFERRED_VERSION_linux-renesas-rt = "6.1%"

# Select the linux version: v6.1
# PREFERRED_VERSION_linux-renesas = "6.1%"
```

### (3) 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
```

Expand disk image.

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

The file of *<wic image>* is listed in the Table 2.

**Table 3. File and directory in the micro SD card**

| Type/Number | Filesystem | Contents |
|---|---|---|
| Primary #1 | VFAT | FlashWriter-smarc-rzg3s.mot<br>bl2_bp_spi-smarc-rzg3s.srec<br>bl2_bp_mmc-smarc-rzg3s.srec<br>bl2_bp_esd-smarc-rzg3s.srec<br>bl2_bp_spi-smarc-rzg3s.bin<br>bl2_bp_mmc-smarc-rzg3s.bin<br>bl2_bp_esd-smarc-rzg3s.bin<br>fip-smarc-rzg3s.srec<br>fip-smarc-rzg3s.bin |
| Primary #2 | Ext4 | ./<br>├── bin<br>├── boot<br>│ ├── Image<br>│ └── r9a08g045s33-smarc.dtb<br>├── dev<br>├── etc<br>├── home<br>├── lib<br>├── media<br>├── mnt<br>├── proc<br>├── run<br>├── sbin<br>├── sys<br>├── tmp<br>├── usr<br>└── 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)
- 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).

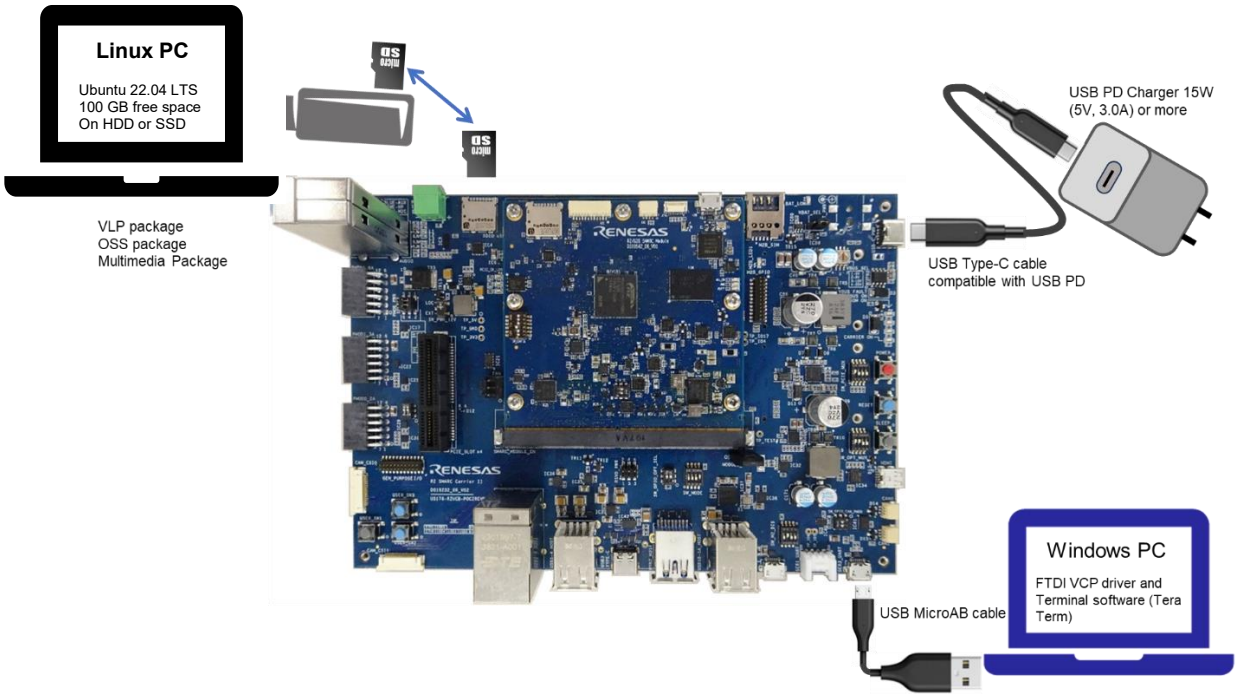



**Figure 2. Operating environment**

### 4.1.1 How to set boot mode and input voltage

Please set the DIP switch “SW_MODE” and the rotary switch “VBUS_SEL” as follows.

- Pins no. 1 to no. 3 of “SW_MODE” are used to control the boot mode of the RZ/G3S.

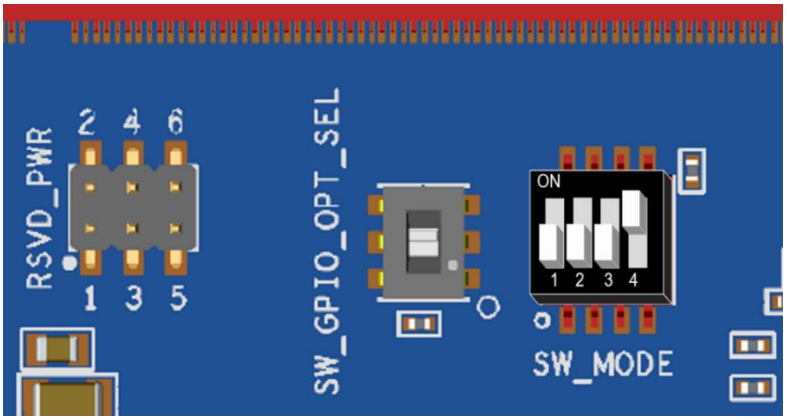



**Figure 3. Location and Default Settings of SW_MODE at the Bottom Center of the RZ SMARC Carrier II**

**Table 4. DIP Switch “SW_MODE“ Settings**

| SW_MODE[1] (BOOT_SEL0#) | SW_MODE[2] (BOOT_SEL1#) | SW_MODE[3] (BOOT_SEL2#) | Function |
|---|---|---|---|
| ON | ON | OFF | eSD boot (3.3-V start-up) |
| OFF | ON | OFF | SCIF download |
| ON | OFF | OFF | eMMC boot (1.8 V) |
| OFF | OFF | OFF | SPI boot (1.8 V) |

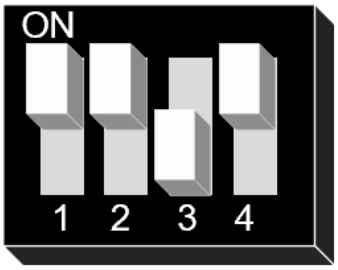



Figure 4. eSD Boot Mode

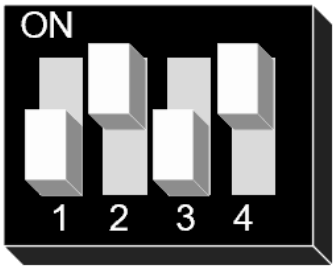



Figure 5. SCIF Download Mode

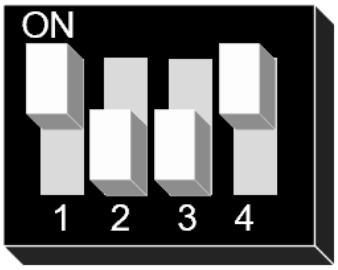



Figure 6. eMMC Boot Mode

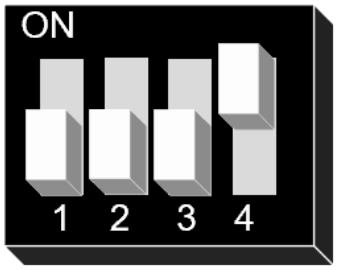



Figure 7. SPI Boot Mode

### 4.1.2 How to set “VBUS_SEL” rotary switch

The RZ SMARC Series Carrier Board II primary power input is from the USB Type-C connected to a power delivery device (not provided).

Check the following switch settings match the minimum power requirements and the capabilities of the USB-C power adapter.

- Pin no. 4 of “SW_MODE” is used to control the minimum input current from the power adapter to 2 A or 4 A.
- “VBUS_SEL” is used to control the minimum input voltage from the power adapter to 5 V, 9 V, 15 V or 20 V.

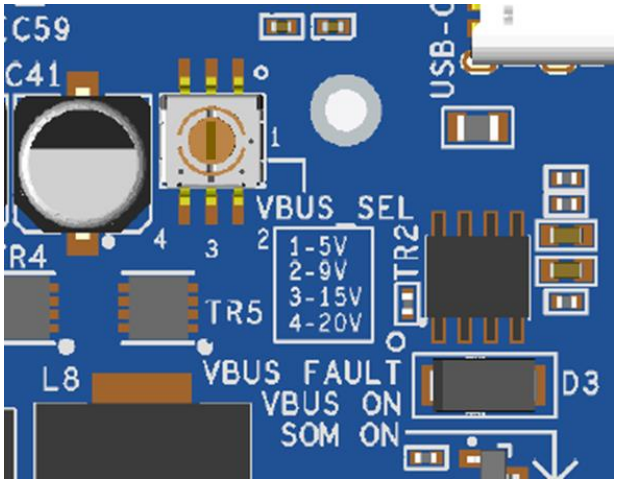



**Figure 8. Location and Default Settings of VBUS_SEL at the Top Right of the RZ SMARC Carrier II**

**Table 5. DIP Switch “VBUS_SEL“ Settings**

| SW_MODE[4] | VBUS_SEL | Min. Voltage | Min. Current | Min. Power |
|---|---|---|---|---|
| ON | 1 | 5 V | 2 A | 10 W |
| ON | 2 | 9 V | 2 A | 18 W |
| ON | 3 | 15 V | 2 A | 30 W |
| ON | 4 | 20 V | 2 A | 40 W |
| OFF | 1 | 5 V | 4 A | 20 W |
| OFF | 2 | 9 V | 4 A | 36 W |
| OFF | 3 | 15 V | 4 A | 60 W |
| OFF | 4 | 20 V | 4 A | 80 W |

### 4.1.3 How to Set “SW_CONFIG” DIP switch

A bank of 6-switches is used to configure the operating modes of the module.

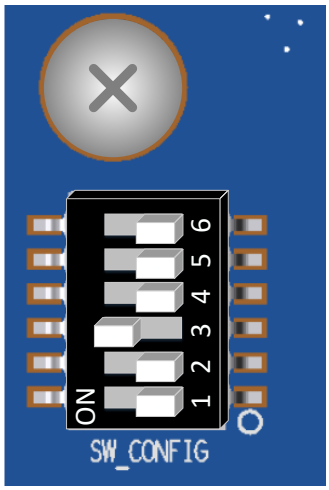



**Figure 9. Location and Default Settings of SW_CONFIG at the Middle Left of the RZ/G3S SMARC Module**

**Table 6. DIP Switch “SW_CONFIG“ Settings**

| SW_CONFIG[x] | Signal | ON | OFF |
|---|---|---|---|
| 6 | RZ_BOOTCPUSEL | Cold boot from Cortex-M33 (w/o FPU) | Cold boot from Cortex-A55 |
| 5 | RZ_MD_CLKS | SSCG is disabled | SSCG is enabled |
| 4 | SW_I3C_VIO_SEL | I3C voltage is 1.2 V | I3C voltage is 1.8 V |
| 3 | SW_SD2_EN | SD2 is disabled | SD2 is enabled |
| 2 | SW_SD0_DEV_SEL | SD0 connected to uSD0 card | SD0 connected to eMMC |
| 1 | RZ_DEBUGEN | JTAG debug is disabled | JTAG debug is enabled |

### 4.1.4 How to set “SW_OPT_MUX” DIP switch

To support legacy modules and provide greater flexibility, the RZ SMARC Carrier II provides multiplexing options for SDIO, I2S and SER0.

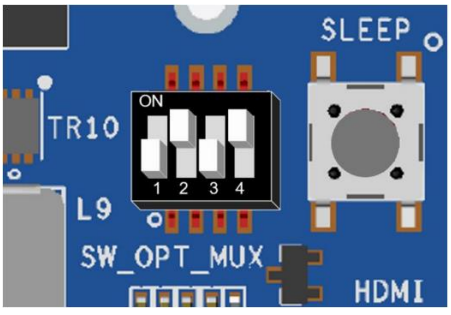



**Figure 10. The Location and the Default Setting of the SW_OPT_MUX**

**Table 7. DIP Switch “SW_OPT_MUX” Settings**

| SW_OPT_MUX [x] | Signals and routed | ON | OFF |
|---|---|---|---|
| 4 | SER0 signals | PMOD1 | M.2 Key E UART |
| 3 | - | | |
| 2 | I2S signals | M.2 Key E interface | External display interface DISP0 |
| 1 | SDIO signals | M.2 Key E interface | uSD on the Carrier |

If you boot Linux, please SW_OPT_MUX [1] is OFF.

### 4.1.5 How to use debug serial (console output)

Connect the USB Type Micro-AB cable to the connector “SER3_UART”.

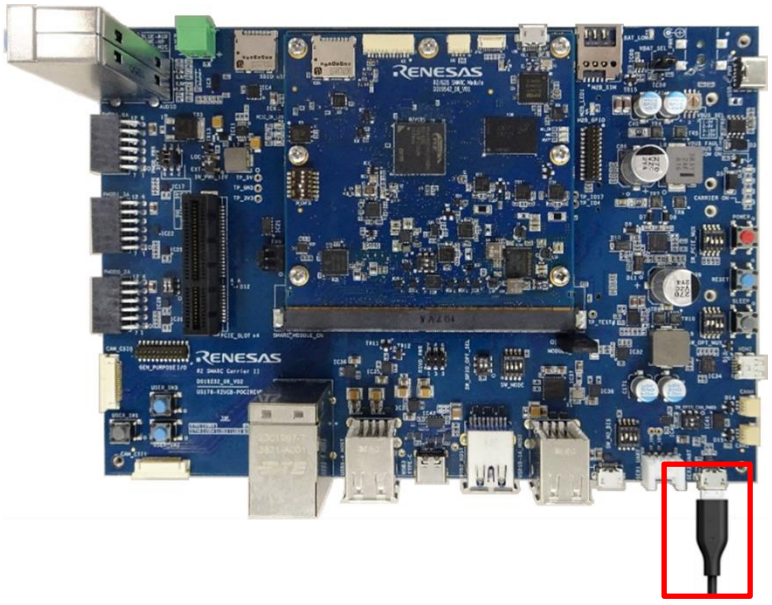


**Figure 11. Connecting console for debug (SER3_UART)**

## 4.2 Startup Procedure

This section describes how to write Flash writer and bootloaders using Windows PC. For describing how to write using Ubuntu PC, please refer to Section **8.5**.

### 4.2.1 Power supply

1. Set the RZ/G3S SMARC EVK boot switch selector (“SW_MODE”) to the selected boot mode. See **How to set boot mode and input voltage** and **How to Set “SW_CONFIG” DIP switch** for details such as other boot mode, DIP switch, and jumper pin settings.
2. Attach the PD capable USB Type-C cable to the USB Type-C port (“USB-C_PWR_IN”).
3. Connect the other end of the Type-C cable to the source that is the USB-C power adapter.
4. Visually check that two LEDs (“VBUS_PWR_ON” and “SOM_PWR_ON”) are illuminated.

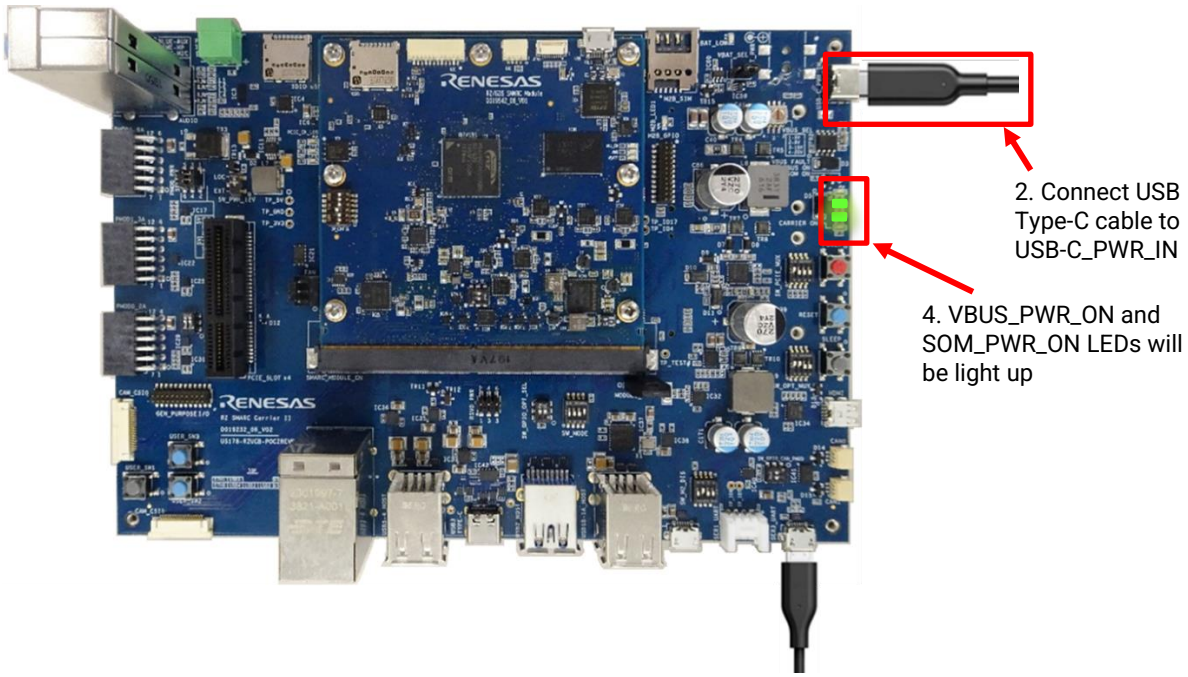
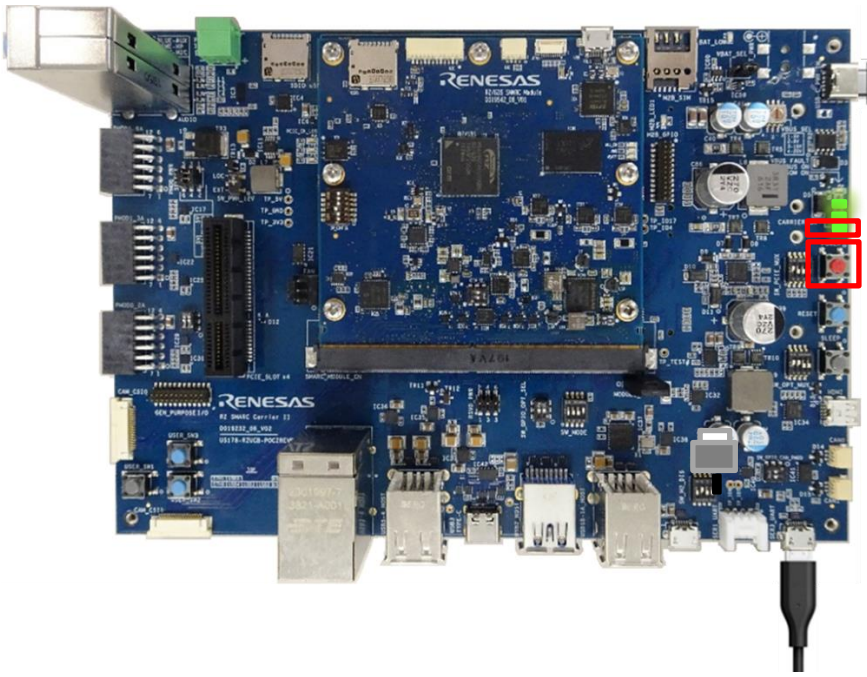



**Figure 12. Connecting Power Supply**

5. Push the “Power” red button for 1 sec and visually check that LED (“CARRIER_PWR_ON”) is illuminated.



5. Press the Red button (PWR)

CARRIER_PWR_ON LED will be light up.

**Figure 13. Power ON**

### 4.2.2 Building files to write

The evaluation boards use the files in the Table 8 as boot loaders. The boot loaders files are generated by following the build instruction. Please refer to Table 2. Once builded copy these files to a PC which runs serial terminal software.

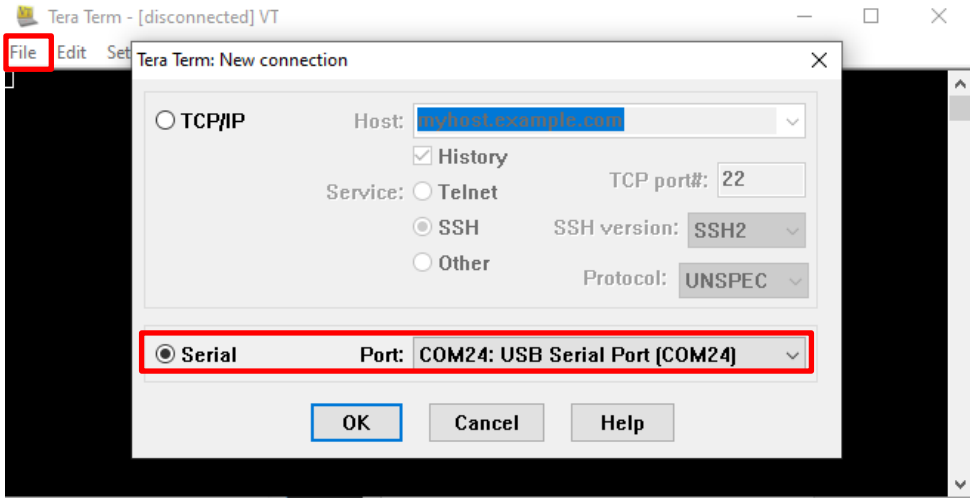
**Table 8. File names of Boot loader**

| Board | File name of Boot loader |
|---|---|
| RZ/G3S Evaluation Board Kit | • bl2_bp_spi-smarc-rzg3s.srec<br>• fip-smarc-rzg3s.srec |

### 4.2.3 Settings

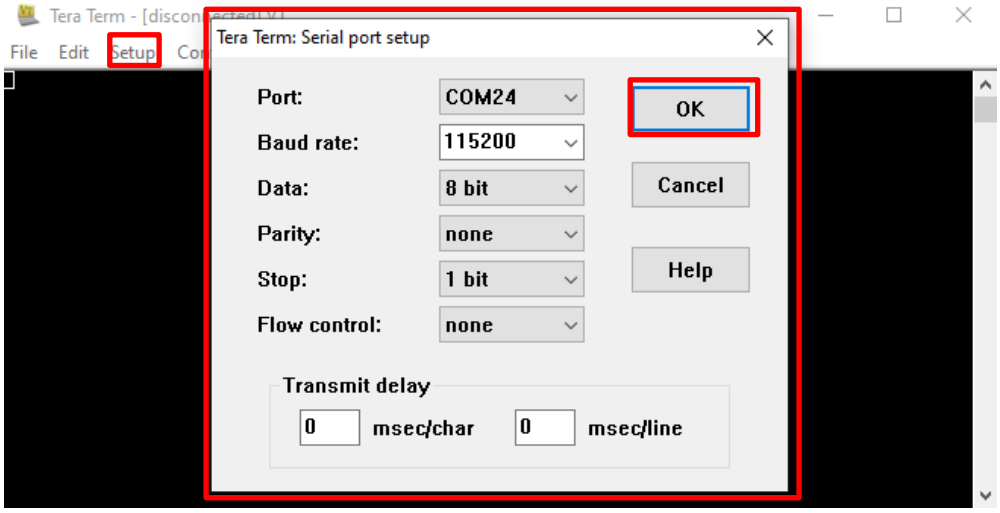
Connect the board to a host PC using the USB serial cable according to the Release Note.

1. To set the board to SCIF download mode, set “SW_MODE" according to the switch setting in Figure 5 of Table 6. DIP Switch “SW_CONFIG“ Settings.
2. Power the RZ/G3S SMARC EVK according to Startup Procedure.
3. Launch the terminal software and select “File” > “New Connection” to set the connection on the software.
4. Select “Setup” > “Serial port” to configure settings related to serial communication protocols on the software. Set the serial communication protocol settings on a terminal software as follows.



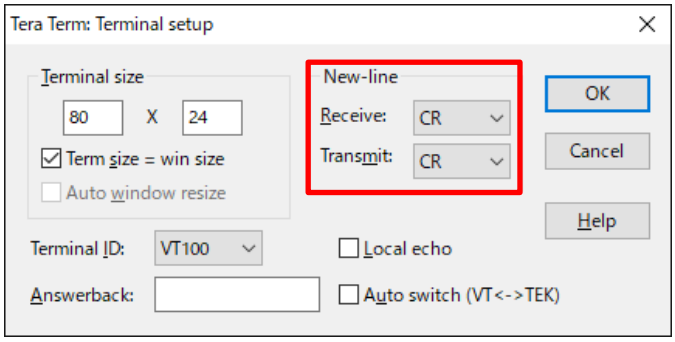

Set the settings about serial communication protocol on a terminal software as below:

- 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”




**(1) To set the board to SCIF Download mode, set the SW_MODE as below:**

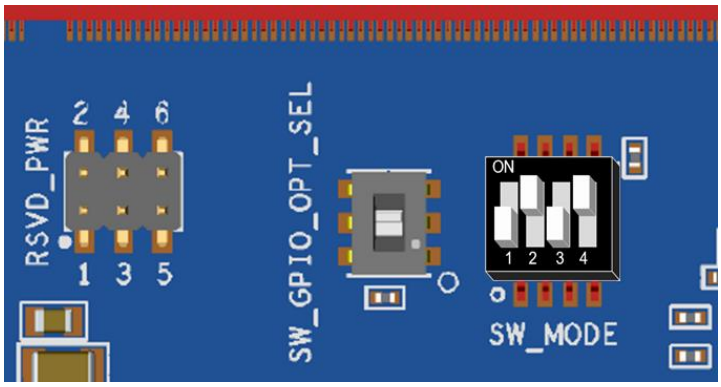



**Table 9. SW_MODE**

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| OFF | ON | OFF | ON |

**(2) After finishing all settings, when pressing the RESET button, the messages below are displayed on the terminal.**

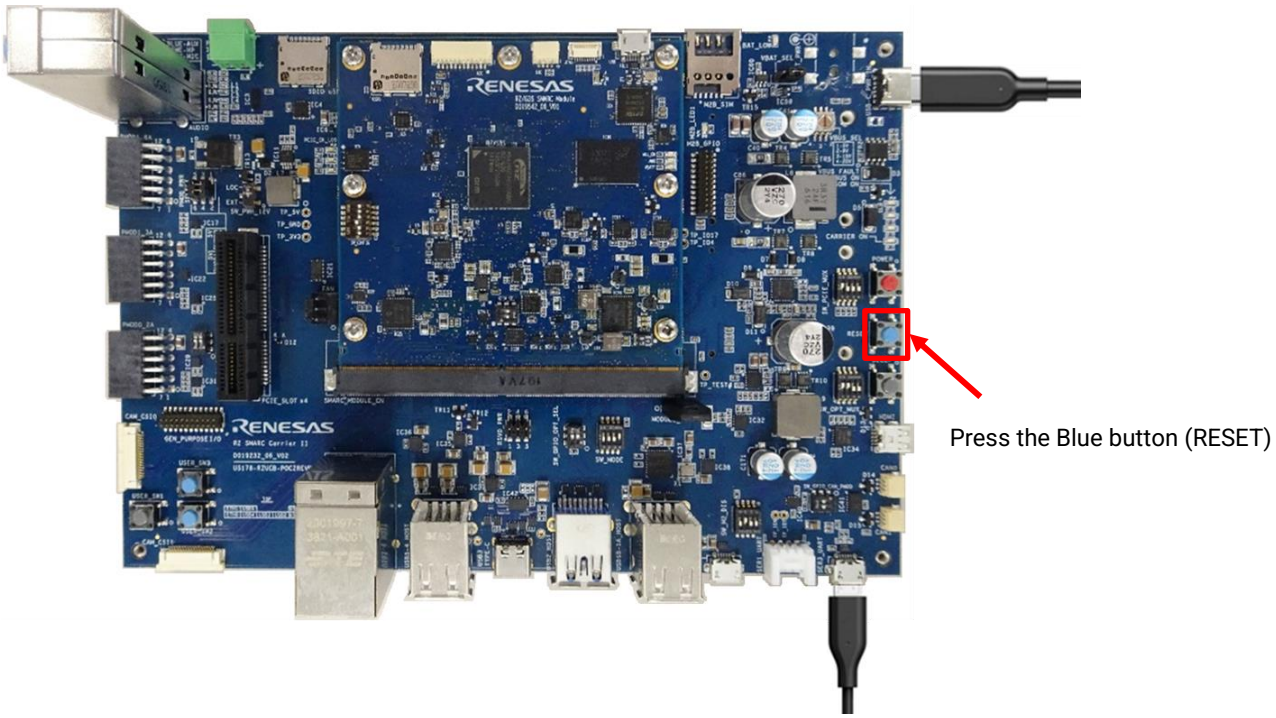



**Figure 14. Press the RESET button**

## 4.3 Download Flash Writer to RAM

Copy the Flash Writer and Bootloader files from the build-generated image directory or from the microSD card where the wic image was deployed, as shown in **Table 3**, to your PC.

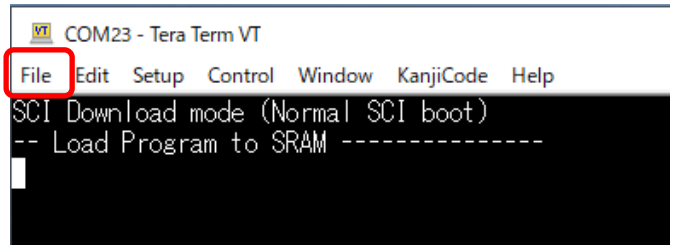
Turn on the power of the board by pressing the PWR button. The messages below are shown on the terminal.

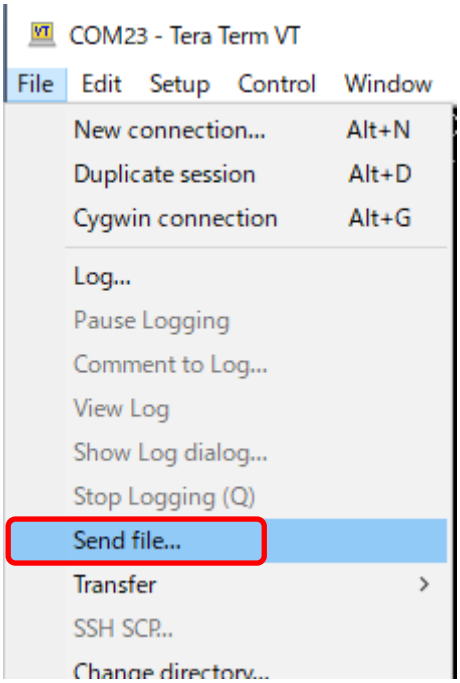
```
SCI Download mode (Normal SCI boot)
-- Load Program to SRAM ---------------
```

Send an image of Flash Writer (FlashWriter-smarc-rzg3s.mot) using terminal software.

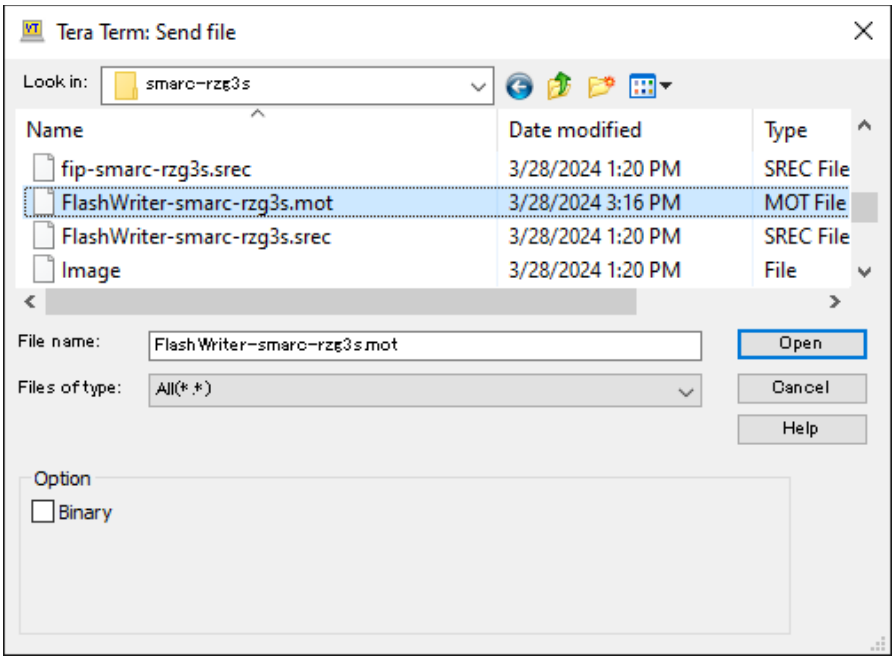
Below is a sample procedure with Tera Term.

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

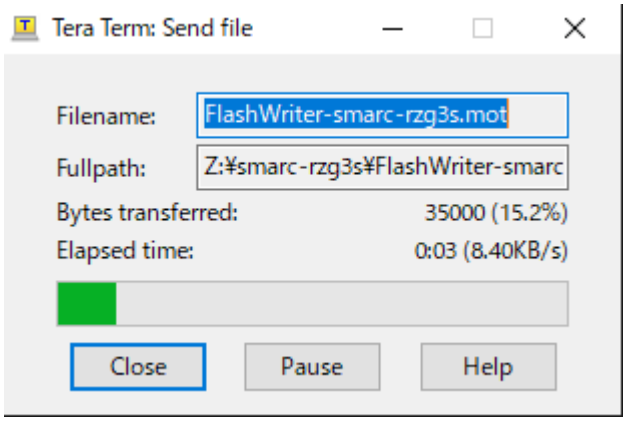



Then, select the image to be send and click “Open” button.




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/G3S Series V0.60 Jan.26,2023
Product Code : RZ/G3S
>
```

## 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 Section **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/G3S Evaluation Board Kit.:

```
>XLS2
===== Qspi writing of RZ/G3 Board Command =============
Load Program to Spiflash
Writes to any of SPI address.
Program size & Qspi Save Address
===== Please Input Program Top Address ============
Please Input : H'a1e00
===== Please Input
Qspi Save Address ===
Please Input : H'0
please send ! ('.' & CR stop load)
```

Send the data of “bl2_bp_spi-smarc-rzg3s.srec” from terminal software after the message “please send !” is shown.

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

```
Erase SPI Flash memory...
Erase Completed
Write to SPI Flash memory.
======= Qspi Save Information =================
SpiFlashMemory
Stat Address : H'00000000
SpiFlashMemory
End Address : H'0001BCCF
================================================
```

```
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/G3 Board Command =============
Load Program to Spiflash
Writes to any of SPI address.
Program size & Qspi Save Address
```

```
===== Please Input Program Top Address ============
Please Input : H’0
===== Please Input Qspi Save Address ===
Please Input : H'60000
please send ! ('.' & CR stop load)
```

Send the data of “fip-smarc-rzg3s.srec” from terminal software after the message “please send !” is shown.

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

```
Erase SPI Flash memory...
Erase Completed
Write to SPI Flash memory.
======= Qspi Save Information =================
SpiFlashMemory Stat Address : H'00060000
SpiFlashMemory End Address : H'0014382E
============================================================
```

```
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 Section **4.2.3**.

Finally, turn off the power of the board by pressing the PWR button.

**Table 10. Address for sending each loader binary file**

| File name | Address to load to RAM | Address to save to ROM |
|---|---|---|
| bl2_bp_spi-smarc-rzg3s.srec | a1e00 | 00000 |
| fip-smarc-rzg3s.srec | 00000 | 60000 |

## 4.5 Change Back to Normal Boot Mode

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

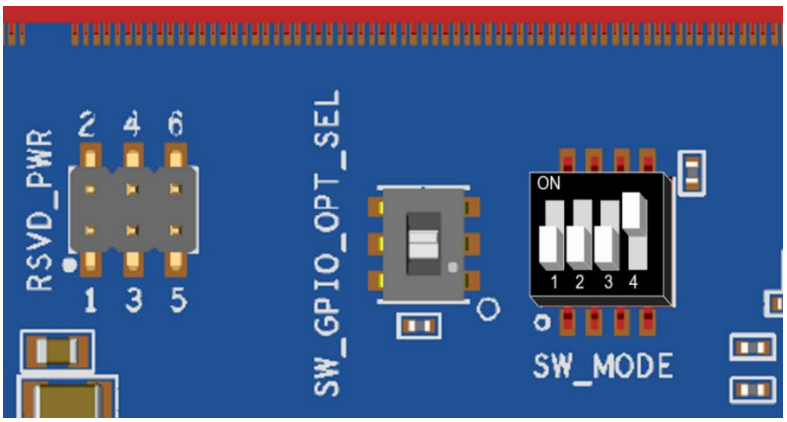



**Table 11. SW_MODE**

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| OFF | OFF | OFF | ON |

Turn on the power of the board by pressing the RESET button.

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

CPU:   Renesas Electronics CPU rev 1.0
Model: smarc-rzg3s
DRAM:  896 MiB
MMC:   sd@11c00000: 0, sd@11c10000: 1, sd@11c20000: 2
Loading Environment from MMC... OK
In:    serial@1004b800
Out:   serial@1004b800
Err:   serial@1004b800
Net:
Error: ethernet@11c30000 address not set.
No ethernet found.

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.

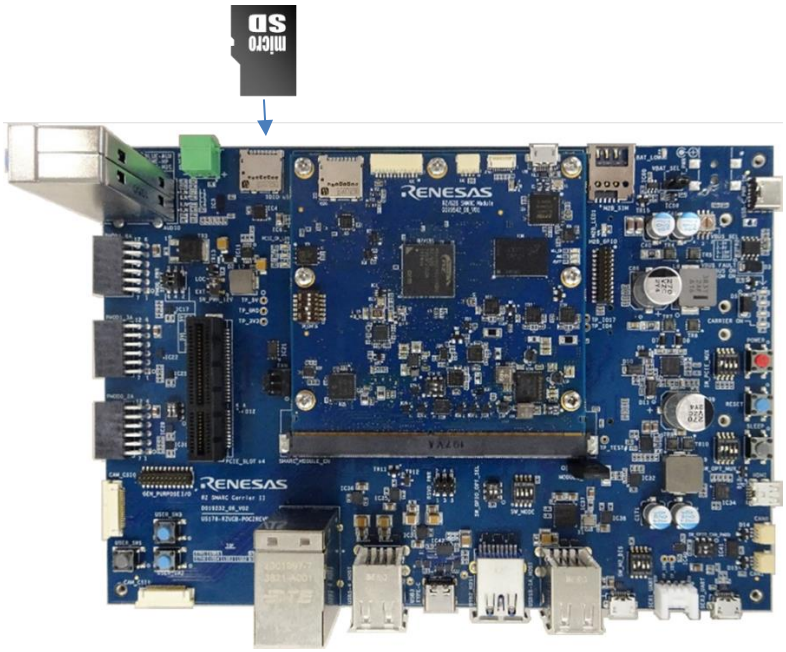



**Figure 15. Set micro SD card to SMARC-EVK**

**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 POWER button 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-rzg3s
DRAM:  896 MiB
MMC:   sd@11c00000: 0, sd@11c10000: 1, sd@11c20000: 2
Loading Environment from MMC... OK
In:    serial@1004b800
Out:   serial@1004b800
Err:   serial@1004b800
Net:
Error: ethernet@11c30000 address not set.
No ethernet found.

Hit any key to stop autoboot:  0
mmc1 is current device
17697280 bytes read in 611 ms (27.6 MiB/s)
```

```
34833 bytes read in 2 ms (16.6 MiB/s)
Moving Image from 0x48080000 to 0x48200000, end=49350000
## Flattened Device Tree blob at 48000000
   Booting using the fdt blob at 0x48000000
   Loading Device Tree to 0000000057ff4000, end 0000000057fff810 ... OK

Starting kernel ...
...
smarc-rzg3s login: root
[   16.378245] audit: type=1006 audit(1600598650.892:2): pid=203 uid=0 old-auid=429496
7295 auid=0 tty=(none) old-ses=4294967295 ses=1 res=1
root@smarc-rzg3s:~#
```

## 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-rzg3s:~# 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 “system-shutdown [1]: powering off” and “reboot: Power down” massage.

# 6. Building the SDK

To build Software Development Kit (SDK), run the commands below after the steps (1) – (6) of the 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 ~/rzg3s_vlp_${PACKAGE_VERSION}/build
$ MACHINE=smarc-rzg3s bitbake core-image-minimal -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 rz-vlp-glibc-x86_64-core-image-minimal-cortexa55-smarc-rzg3s-toolchain-\
5.0.14.sh
```

# 7. Application Building and Running

This chapter explains how to make and run an application for RZ/G3S 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-minimal-cortexa55-smarc-rzg3s-toolchain-\
5.0.14.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:

```
$ sudo sh rz-vlp-glibc-x86_64-core-image-minimal-cortexa55-smarc-rzg3s-toolchain-\
5.0.14.sh
Poky (Yocto Project Reference Distro) SDK installer version 5.0.14
==========================================================================
Enter target directory for SDK (default: /opt/poky/5.0.14):
You are about to install the SDK to "/opt/poky/5.0.14". 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/5.0.14/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/5.0.14/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/5.0.14/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.

Step 4. Store a sample application

The sample application could be written by the following procedure. The application should be stored in the ext3 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/G3S Evaluation Board Kit and start the system. After booting, run the sample application with the following command.

```
smarc-rzg3s login: root
root@smarc-rzg3s:~# /usr/bin/linux-helloworld

Hello World
root@smarc-rzg3s:~#
```

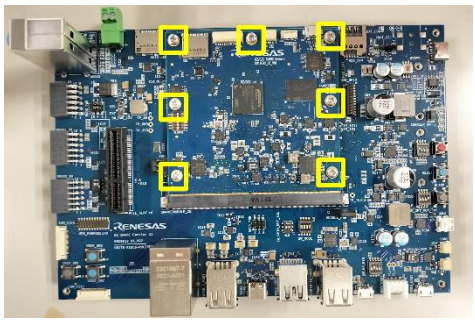
Note: Refer to the start-up guide for the method of how to boot the board and system.

# 8. Appendix

## 8.1 How to replace the SMARC Module Board

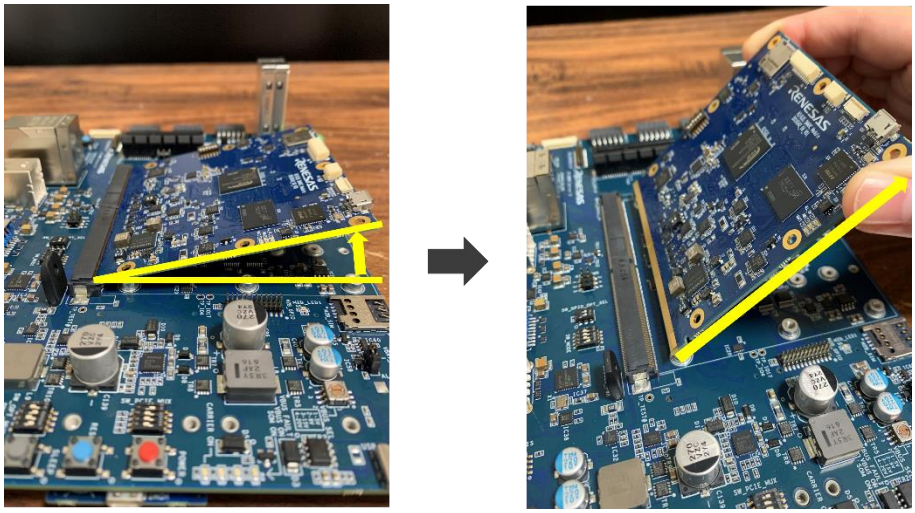
Please be careful when replacing the board as follows.

1. Remove the four screws.

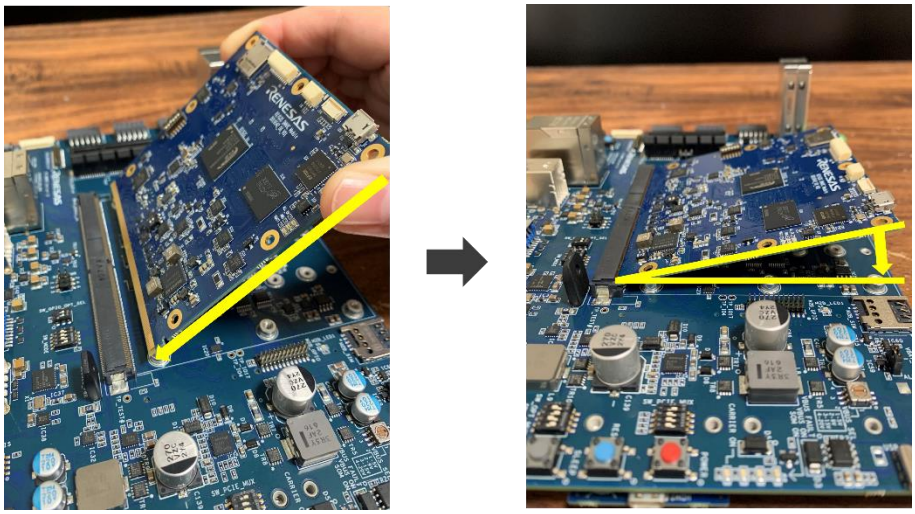




2. Remove the screw and the board will stand up at an angle. Slide it out.



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



## 8.2 How to boot from eMMC

In this section, the steps to boot from eMMC are described.

### 8.2.1 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 12**, and corresponding bootloader files and specified address information are depending on each target board as described in **Table 13**.

After booting the Flash Writer (Refer to 4.3Download Flash Writer to RAM),

Then “em_secsd” to modify EXT_CSD register.

For example, this part describes how to modify EXT_CSD register and write boot loader files in the case of RZ/G3S Evaluation Board Kit:

```
>em_secsd
Please Input EXT_CSD Index(H'00 - H'1FF) : b1
EXT_CSD[B1] = 0x02
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] = 0x09
Please Input Value(H'00 - H'FF) : 8
EXT_CSD[B3] = 0x08
>
```

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.

```
>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 : a1e00
Work RAM (H'00020000-H'000FFFFF) Clear....
please send ! ('.' & CR stop load)
```

Send the data of “bl2_bp_mmc-smarc-rzg3s.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 :300
Please Input Program Start Address : 0
Work RAM(H'00020000-H'000FFFFF) Clear....
please send ! ('.' & CR stop load)
```

Send the data of "fip-smarc-rzg3s.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!
```

**Table 12. Address of EXT_CSD register of eMMC for eMMC boot**

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

**Table 13. Address for sending each loader binary file for eMMC boot**

| File name | Partition to save to eMMC | Address to save to eMMC | Address to load to RAM |
|---|---|---|---|
| bl2_bp_mmc-smarc-rzg3s.srec | 1 | 00000001 | a1E00 |
| fip-smarc-rzg3s.srec | 1 | 00000300 | 00000 |

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

### 8.2.2 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 microSD card.

Copy a kernel image, a device tree file and rootfs which are listed in the Table 2 to the second partition of the microSD card.

```
$ cd /media/user/rootfs/home/root/
$ sudo cp $WORK/build/tmp/deploy/images/smarc-rzg3s/<Linux kernel> ./
$ sudo cp $WORK/build/tmp/deploy/images/smarc-rzg3s/<Device tree> ./
$ sudo cp $WORK/build/tmp/deploy/images/smarc-rzg3s/<root filesystem> ./
$ cd $WORK
$ sudo umount /media/user/rootfs
```

### 8.2.3 Setting U-boot and writing rootfs to eMMC

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

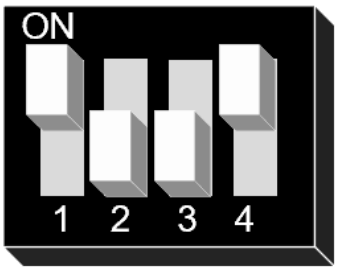



**Table 14. "SW_MODE"**

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| ON | OFF | OFF | ON |

Refer to 4.4 Write the Bootloader and set the u-boot environment variables (please ignore switch settings). Then, turn on the board by pressing the PWR button.

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

```
root@smarc-rzg3s:~# 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-rzg3s:~#
```

Format eMMC.

```
root@smarc-rzg3s:~# mkfs.ext4 /dev/mmcblk0p1
root@smarc-rzg3s:~# mkfs.ext4 /dev/mmcblk0p2
```

Write the rootfs, the kernel and the device tree.

```
root@smarc-rzg3s:~# mount /dev/mmcblk0p1 /mnt/
root@smarc-rzg3s:~# cp Image /mnt/
root@smarc-rzg3s:~# cp r9a08g045s33-smarc.dtb /mnt/
root@smarc-rzg3s:~# umount /dev/mmcblk0p1
root@smarc-rzg3s:~# mount /dev/mmcblk0p2 /mnt/
root@smarc-rzg3s:~# tar xf /home/root/core-image-minimal-smarc-rzg3s.rootfs.tar.bz2 -C
 /mnt/
root@smarc-rzg3s:~# umount /dev/mmcblk0p2
```

### 8.2.4 Setting U-boot for eMMC boot

Reset the board by pressing the RESET button.

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

CPU:   Renesas Electronics CPU rev 1.0
Model: smarc-rzg3s
DRAM:  896 MiB
MMC:   sd@11c00000: 0, sd@11c10000: 1
Loading Environment from MMC... OK
In:    serial@1004b800
Out:   serial@1004b800
Err:   serial@1004b800
Net:   No ethernet found.

Hit any key to stop autoboot:  0
=>
```

Set environment variables to boot from eMMC.

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

Please reset the board again for eMMC boot.

## 8.3 How to boot from eSD

This section describes the steps to boot from the micro-SD card inserted in the EVK.

RZ/G3S can boot from eSD. To make evaluation easier, RZ/G3S SMARC EVK has a micro-SD card slot instead of an eSD device, hence the kit can boot from the micro-SD card as well. Note that there are some differences in specifications between eSD and SD, some kinds of SD cards may not be usable for this purpose.

Note) This boot mode is available for V03 or newer versions of RZ/G3S SMARK EVK. Please check the printed version number at the following locations on the SOM module.



### 8.3.1 Prepare micro SD card

Prepare the micro SD card using the wic image file.

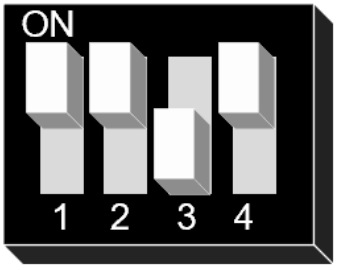
Two files (bl2_bp_esd-smarc-rzg3s.bin and fip-smarc-rzg3s.bin) are used for boot from eSD.

**Table 15. File and directory in the micro SD card**

| Type/Number | Filesystem | Contents |
|---|---|---|
| Primary #1 | VFAT | FlashWriter-smarc-rzg3s.mot<br>bl2_bp_spi-smarc-rzg3s.srec<br>bl2_bp_mmc-smarc-rzg3s.srec<br>bl2_bp_esd-smarc-rzg3s.srec<br>bl2_bp_spi-smarc-rzg3s.bin<br>bl2_bp_mmc-smarc-rzg3s.bin<br>bl2_bp_esd-smarc-rzg3s.bin<br>fip-smarc-rzg3s.srec<br>fip-smarc-rzg3s.bin |
| Primary #2 | Ext4 | ./<br>├── bin<br>├── boot<br>│ ├── Image<br>│ └── r9a08g045s33-smarc.dtb<br>├── dev<br>├── etc<br>├── home<br>├── lib<br>├── media<br>├── mnt<br>├── proc<br>├── run<br>├── sbin<br>├── sys<br>├── tmp<br>├── usr<br>└── var |

### 8.3.2 Set SMARC EVK board for eSD boot

Change the switches to eSD boot on the carrier board acting on SW_MODE (SW_MODE [1] ON, SW_MODE [2] ON, SW_MODE [3] OFF, SW_MODE [4] ON)



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

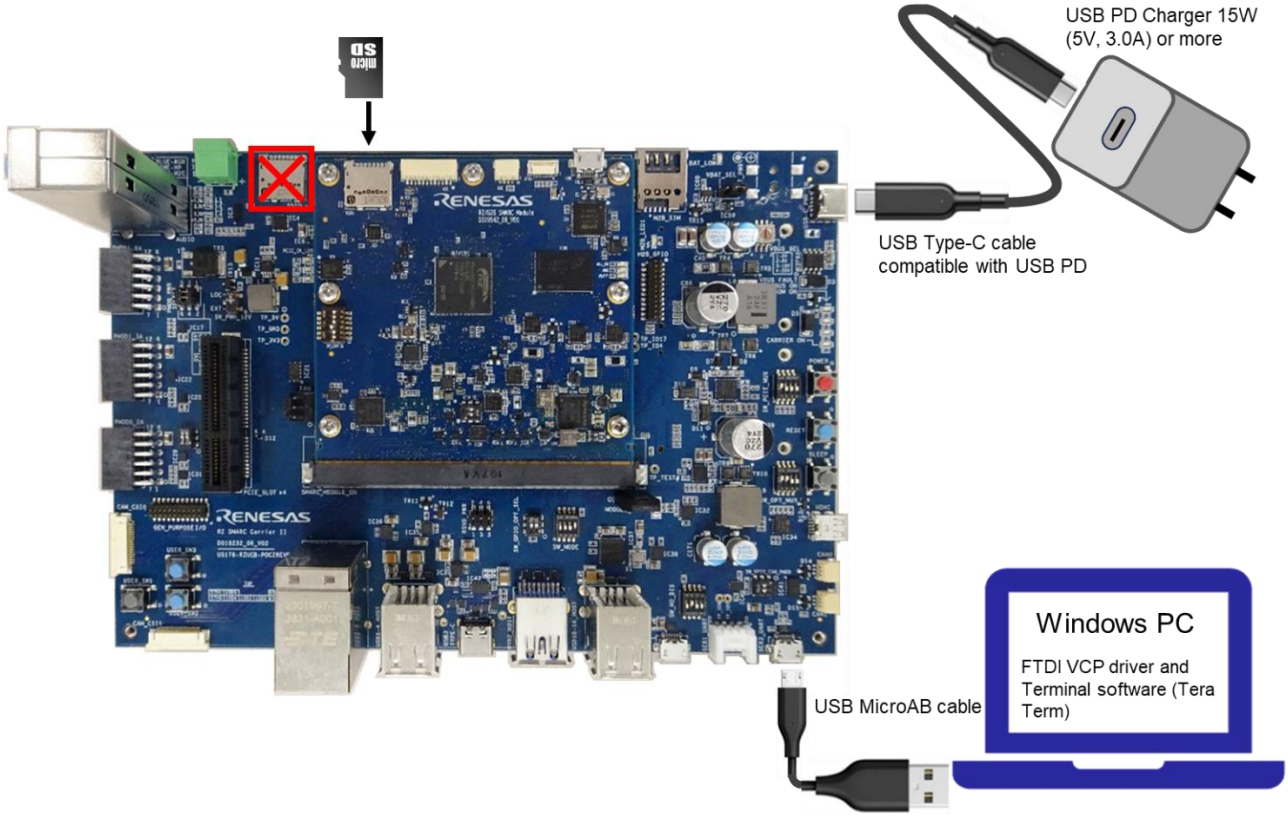



**Figure 16. Operating environment for eSD boot**

### 8.3.3 Power on and boot

After pressing the red button (POWR) to turn on the power. And press the blue button (RESET), Linux will be booted from eSD.

```
NOTICE: BL2: v2.10.5(release):181ce7f-dirty
NOTICE: BL2: Built : 14:17:37, Jul 10 2026
NOTICE: BL2: SYS_LSI_MODE: 0x13241
NOTICE: BL2: SYS_LSI_DEVID: 0x185e0447
NOTICE: BL2: Booting BL31
NOTICE: BL31: v2.10.5(release):181ce7f-dirty
NOTICE: BL31: Built : 14:17:37, Jul 10 2026


U-Boot 2024.07 (Jun 09 2026 - 11:56:56 +0000)

CPU: Renesas Electronics R9A08G045S rev 2.0
Model: smarc-rzg3s
DRAM: 896 MiB
Core: 33 devices, 17 uclasses, devicetree: separate
MMC: sd@11c00000: 0, sd@11c10000: 1, sd@11c20000: 2
Loading Environment from MMC... Reading from MMC(0)... *** Warning - bad CRC, using de
fault environment

In: serial@1004b800
Out: serial@1004b800
Err: serial@1004b800
Net:
Error: ethernet@11c30000 No valid MAC address found.
No ethernet found.

Hit any key to stop autoboot: 0
## Resetting to default environment
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 6.1.141-cip43-yocto-standard (oe-user@oe-host) (aarc
 :
 :
Poky (Yocto Project Reference Distro) 5.0.11 smarc-rzg3s ttySC0

smarc-rzg3s login: root
root@smarc-rzg3s:~#
```

## 8.4 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 (2) 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) Start the build

Process the step 2.2(6) to start the build to get the new image that contains the docker.

#### (4) 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-rzg3s:~# systemctl status docker.service
root@smarc-rzg3s:~# docker version
```

#### (5) Hello world

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

```
root@smarc-rzg3s:~# docker pull hello-world
root@smarc-rzg3s:~# docker run hello-world
Hello from Docker!
```

## 8.5 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                                                |
| FlashWriter-smarc-rzg3s.mot                                              |
| bl2_bp_spi-smarc-rzg3s.srec                                              |
| fip-smarc-rzg3s.srecc                                                    |
|              ( 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 "FlashWriter-smarc-rzg3s.mot”                |
|                                                             |
|102.3 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 Section 4.4 Write the Bootloader. Upload with the “ascii” command in the same way as a Flash Writer file.

## 8.6 Device drivers

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

| File | Description |
|---|---|
| RTK0EF0045Z9006AZJ-v4.0.x.zip | BSP Manual Set for RZ/G2L, RZ/G3S, RZ/Five, RZ/V2L, and RZ/V2N Group |

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

**Table 16. Support device drivers**

| Device Driver | Documents |
|---|---|
| Kernel Core | r01us0468ej0xxx-rz_Kernel_Core_UME.pdf |
| GPIO | r01us0488ej0xxx-rz_GPIO_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 |
| I2C Bus Interface | r01us0477ej0xxx-rz_I2C_UME.pdf |
| I3C Bus Interface | r01us0691ej0102-rz_I3C_UME.pdf |
| Renesas Serial Peripheral Interface (RSPI) | r01us0481ej0xxx-rz_RSPI_UME.pdf |
| SPI Multi IO Bus Controller | r01us0693ej0102-rz_xSPI_UME.pdf |
| RS-CANFD Interface | r01us0478ej0xxx-rz_CANFD_UME.pdf |
| Gigabit Ethernet Interface | r01us0475ej0xxx_rz_Gigabit_Ethernet_UME.pdf |
| USB 2.0 Host | r01us0485ej0xxx-rz_USB2.0_Host_UME.pdf |
| USB 2.0 Function | r01us0491ej0xxx-rz_USB2.0_Function_UME.pdf |
| SD/MMC Host Interface | r01us0474ej0xxx-rz_SD_MMC_UME.pdf |
| Serial Sound Interface | r01us0490ej0115-rz_Audio_UME.pdf |
| A/D Converter | r01us0487ej0xxx-rz_AD_Converter_UME.pdf |
| Thermal Sensor Unit (TSU) | r01us0486ej0xxx-rz_Thermal_Sensor_UME.pdf |
| Power Management | r01us0605ej0xxx-rz_Power_Management_UME.pdf |
| PCI Express Interface | r01us0729ej0102-rz_PCIe_UME.pdf |
| GTM and CMTW | r01us0700ej0xxx-rz_OSTM_UME.pdf |
| Realtime Clock (RTC) | r01us0701ej0xxx-rz_RTC_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 | Nov. 30, 2025 | - | First edition issued. |
| 1.01 | Sep. 30, 2026 | - | Update to v4.0.2 |

## Website and Support

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

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

All trademarks and registered trademarks are the property of their respective owners.