SMARC EVK of RZ/G3S

Linux Start-up Guide

R01US0645EJ0103
Rev.1.03
May 31, 2024

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 3.0.6 or later. (hereinafter referred to as “VLP/G”)

Contents

1. Environment 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
3.1 Write files to the microSD card (used wic image) 11
3.2 Write files to the microSD card (not used wic image) 12
4. Reference Board Setting 16
4.1 Preparation of Hardware and Software 16
4.1.1 How to set boot mode and input voltage 17
4.1.2 How to set “VBUS_SEL” rotary switch 19
4.1.3 How to Set “SW_CONFIG” DIP switch 20
4.1.4 How to use debug serial (console ouput) 20
4.2 Startup Procedure 21
4.2.1 Power supply 21
4.2.2 Building files to write 23
4.2.3 Settings 23
4.3 Download Flash Writer to RAM 26
4.4 Write the Bootloader 28
4.5 Change Back to Normal Boot Mode 30
5. Booting and Running Linux 31
5.1 Power on the board and Startup Linux 31
5.2 Shutdown the Board 33
6. Building the SDK 34
7. Application Building and Running 35
7.1 Make an application 35
7.1.1 How to extract SDK 35
7.1.2 How to build Linux application 35
7.2 Run a sample application 37
8. Appendix 38
8.1 How to replace the SMARC Module Board 38
8.2 How to boot from eMMC 39
8.2.1 Writing Bootloader for eMMC Boot 39
8.2.2 Create a microSD card to boot linux for eMMC boot 40
8.2.3 Setting U-boot and Writing rootfs to eMMC 41

8.2.4 Setting U-boot for eMMC boot 43
8.3 Docker 44
8.4 Booting Setup with Ubuntu PC 45
8.5 Device drivers 47

9. Revision History 48

Website and Support 49

1. Environment Requirement

The environment for building the Board Support Package (hereinafter referred to as “VLP”) is listed inTable 1. Please refer to the below documents 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 the serial terminal interface with software such as TeraTerm.

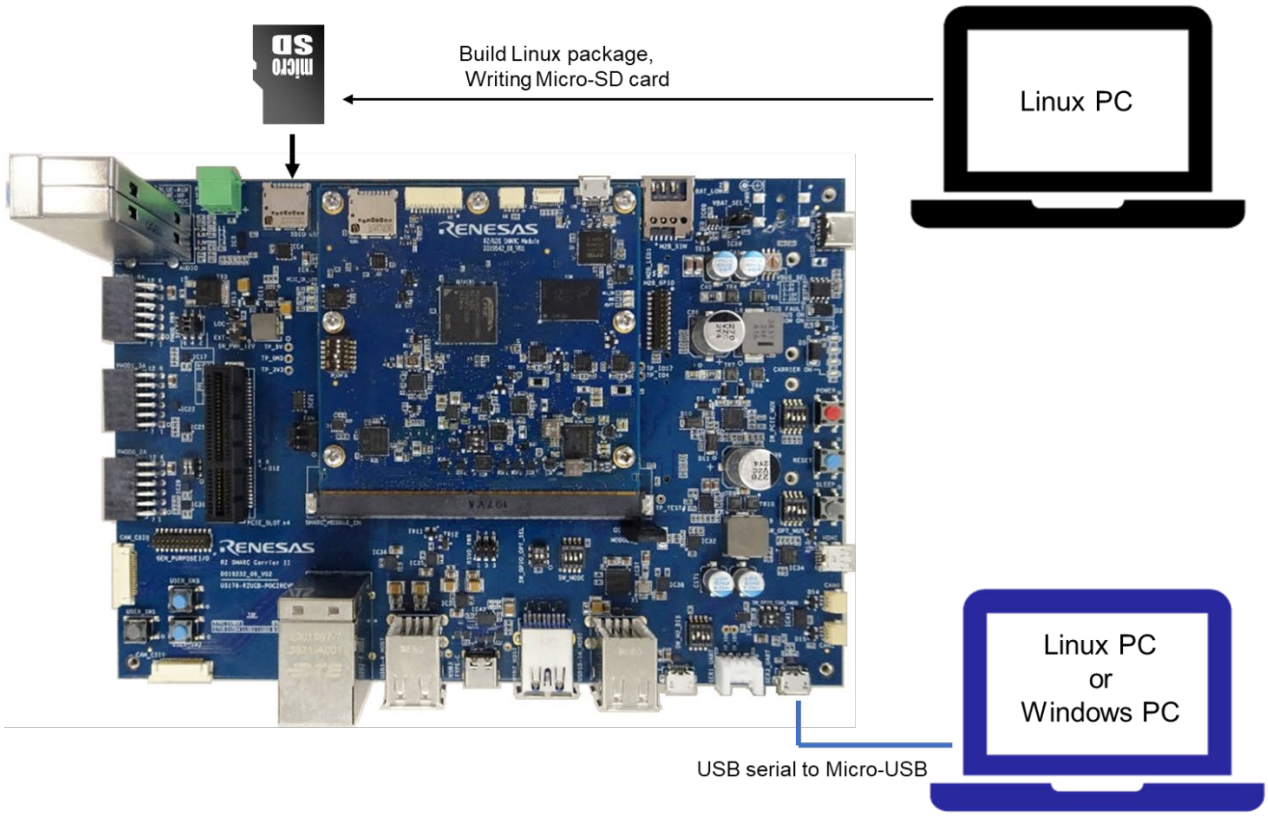


Figure 1. Recommend environment.

Table 1. Equipment and Software for Developing Environments of RZ/G3s Linux Platform

Equipment	Description
Linux Host PC	Used as build/debug environment 100GB free space on HDD or SSD is necessary
OS	Ubuntu 20.04 LTS 64 bit OS must be used. 20.04 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 Please install the VCP Driver corresponding to the target board.
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. Note that use a micro-SDHC card for the flash writer.

2. Build Instructions

2.1 Required Host OS

⚠ The VLP/G is only built in **Ubuntu 20.04**

Ubuntu 20.04 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 22.04 is not supported.

However, you can build the BSP inside a Docker container running Ubuntu 20.04. Instructions for creating this Docker container can be found here:

- https://github.com/renesas-rz/docker_setup

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-get install gawk wget git-core diffstat unzip texinfo gcc-multilib \
build-essential chrpath socat cpio python 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/3.1.31/brief-yoctoprojectqs/brief-yoctoprojectqs.html

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

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

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

(1) 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 ~/rzg_vlp_<package version>
$ cd ~/rzg_vlp_<package version>
$ cp ../<package download directory>/*.zip .
$ unzip ./RTK0EF0045Z95001AZJ-<package version>.zip
$ tar zxvf ./RTK0EF0045Z95001AZJ-<package version>/rzg_vlp_<package version>.tar.\
gz
```

Note) Please note that your build environment must have 100GB of free hard drive space in order to complete the minimum build. The Yocto BSP build environment is very large. Especially in case you are using a Virtual Machine, please check how much disk space you have allocated for your virtual environment.
<package version>: e.g v3.0.6

(2) 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-rzg3s/docs/template/conf/ source \
poky/oe-init-build-env build
```

(3) Add layers (Optional)

- **Docker**: Please run the below commands 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
```

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

Run the commands below. This step is not mandatory and able to go to the step (5) 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/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 "~/rzg_vlp_<package version>/build/conf/local.conf" as below:

```
BB_NO_NETWORK = "1"
```

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

(5) 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=smarc-rzg3s bitbake core-image-<target>
```

<target> can be selected in below. Please refer to the Table 2 for supported image details.

- core-image-minimal
- core-image-bsp

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

```
NOTE: Tasks Summary: Attempted 3795 tasks of which 8 didn't need to be rerun and all s
ucceeded.
```

All necessary files listed in Table 3 will be generated by the bitbake command and will be located in the **`build/tmp/deploy/images`** directory.

BSP can build a few types of images listed in Table 2.

Table 2. Supported images of BSP

Image name	Purpose
core-image-minimal	Minimal set of components
core-image-bsp	Minimal set of components plus audio support and some useful tools

Table 3. Image files for RZ/G3S

RZ/G3S	**Linux kernel**	Image-smarc-rzg3s.bin
	Device tree file	Image-r9a08g045s33-smarc.dtb
	root filesystem	*<image name>*-smarc-rzg3s.tar.bz2
	Boot loader	• bl2_bp_spi-smarc-rzg3s.srec • fip-smarc-rzg3s.srec
	Flash Writer	FlashWriter-smarc-rzg3s.mot
	SD image	core-image-*<image name>* -smarc-rzg3s.wic.gz core-image-*<image name>* -smarc-rzg3s.wic.bmap

2.3 Notes

(1) 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 the step (3) of the section 2.2, create a new working directory and prepare the new build environment. Note that not doing this may cause a build error.

(2) 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 the step (3) of the section 2.2, create a new working directory and prepare the new build 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 BSP, the packages will be replaced as below in the Table 4.

Table 4. Versions of all Buster Debian packages

Package	Buster Debian	Bullseye Debian
attr	2.4.48	2.4.48
busybox	1.30.1	1.30.1
coreutils	6.9	6.9
gcc	8.3.0	9.3.1
glib-2.0	2.58.3	2.62.2
glibc	2.28	2.31
kbd	2.0.4	2.2.0
libgcrypt	1.8.4	1.8.5
openssh	7.9p1	8.2p1
perl	5.30.1	5.30.1

(3) WIC image

The name “WIC” is derived from OpenEmbedded Image Creator (oeic). It includes image that system can boot it in hardware device.

- 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

- There are 2 files *wic.bmap and *wic.gz in deploy folder after building successfully. Example:
  - core-image-minimal-smarc-rzg3s.wic.bmap
  - core-image-minimal-smarc-rzg3s.wic.gz

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
```

Expand disk image.

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

The file nemes of *<wic image>* is listed in the Table 3.

Table 5. File and directory in the micro SD card

Type/Number	Size	Filesystem	Contents
Primary #1	500MB (minimum 128MB)	FAT32	FlashWriter-smarc-rzg3s.mot bl2_bp_spi-smarc-rzg3s.srec fip-smarc-rzg3s.srec
Primary #2	All remaining	Ext4	./ ├── bin ├── boot │ ├── Image │ ├── Image-5.10.145-cip17-yocto-standard │ └── Image-r9a08g045s33-smarc.dtb ├── dev ├── etc ├── home ├── lib ├── media ├── mnt ├── proc ├── run ├── sbin ├── sys ├── tmp ├── usr └── var

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 Table 6.

Table 6. Partitions of microSD card

Type/Number	Size	Filesystem	Contents
Primary #1	500MB (minimum 128MB)	FAT32	Boot loader Flash Writer
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):
First sector (2048-62333951, default 2048):
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 the step 4, please unmount them according to the 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/smarc-rzg3s/<root filesystem>
```

Copy Boot loader and Flash Writer to the first partition

```
$ cp $WORK/build/tmp/deploy/images/smarc-rzg3s/<Flash Writer> /media/user/A299-E6A6
$ cp $WORK/build/tmp/deploy/images/smarc-rzg3s/<Boot loader> /media/user/A299-E6A6
```

The file nemes of *<root filesystem>*, *< Flash Writer>* and *< Boot loader>* are listed in the Table 3.

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).

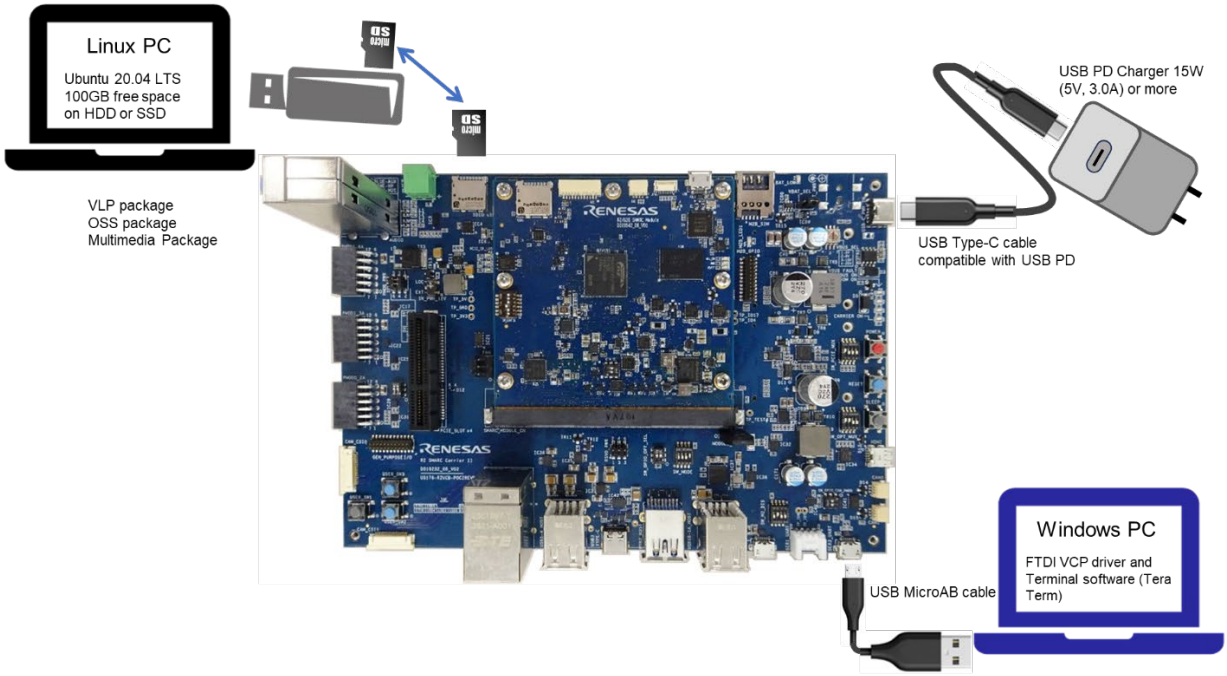


Figure 2. Operating environment

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.

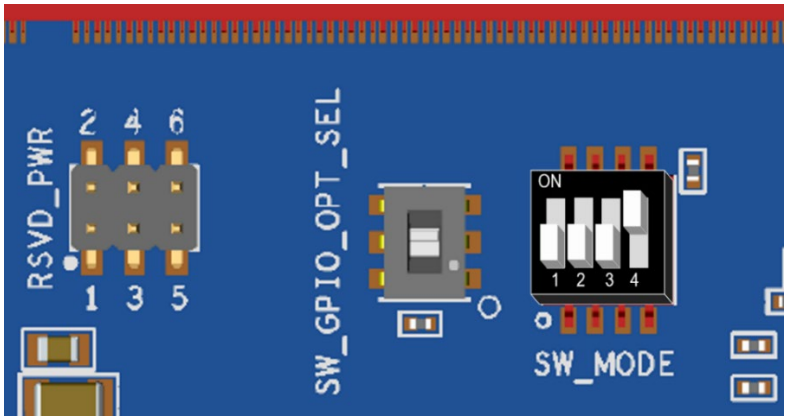


Figure 3. Location and Default Settings of SW_GPIO_OPT_SEL at the Bottom Center of the RZ SMARC Carrier II

Table 7. 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	QSPI boot (1.8 V)

Currently we support 3 modes in 4 modes: SCIF Download mode, QSPI Boot mode and eMMC Boot mode. eSD Boot mode will be supported by the future update.

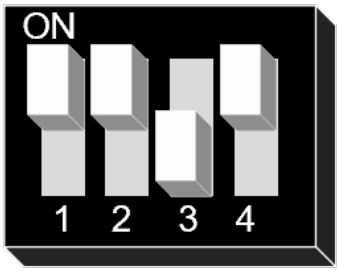


Figure 4. eSD Boot Mode

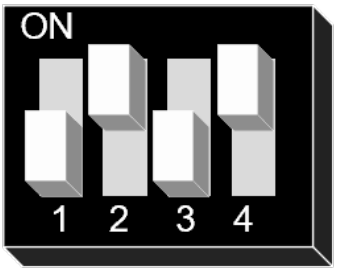


Figure 5. SCIF Download Mode

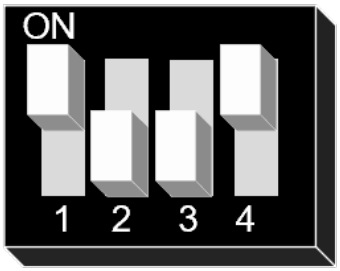


Figure 6. eMMC Boot Mode

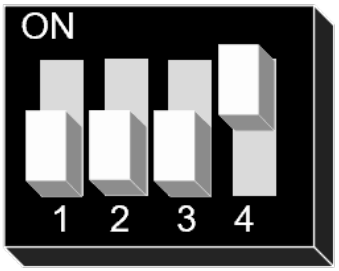


Figure 7. SPI Boot Mode

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.

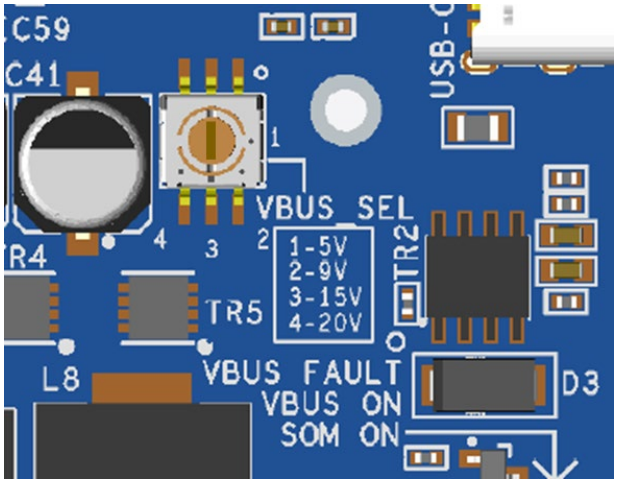


Figure 8. Location and Default Settings of VBUS_SEL at the Top Right of the RZ SMARC Carrier II

Table 8. 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.

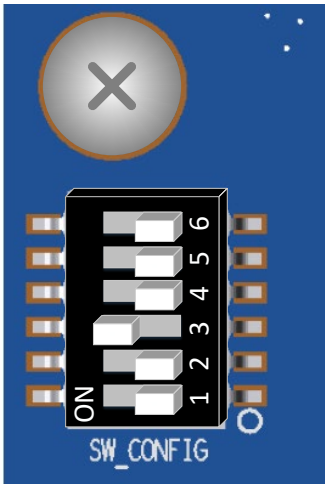


Figure 9. Location and Default Settings of SW_CONFIG at the Middle Left of the G3S SMARC Module

Table 9. 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 use debug serial (console ouput)

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

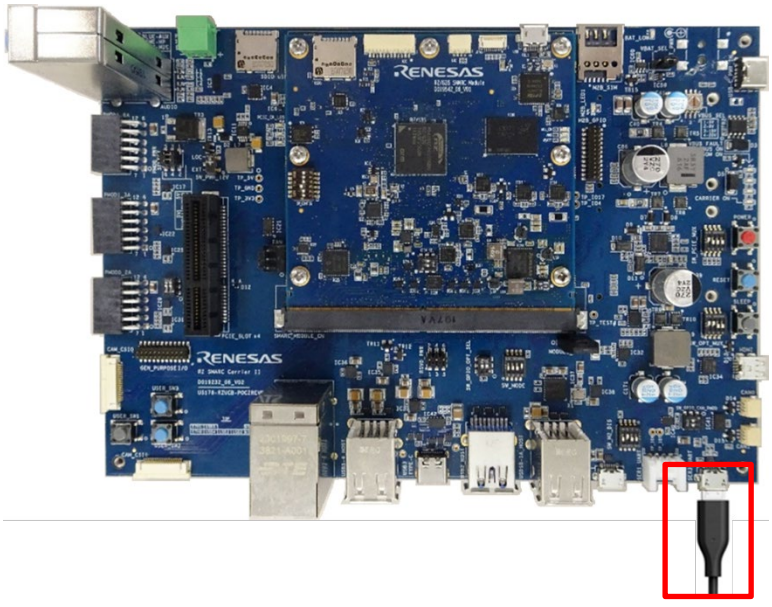


Figure 10. Connecting console for debug (SER3_UART)

4.2 Startup Procedure

This section describes how to write Flash writer and bootloaders using Windows PC. For describes how to write using Ubuntu PC, please refer to 8.4.

4.2.1 Power supply

1. Set the G3S SMARC EVK boot switch selector (“SW_MODE”) to the selected boot mode. See Setting Boot Mode and Input Voltage and Board Configuration and Status 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.

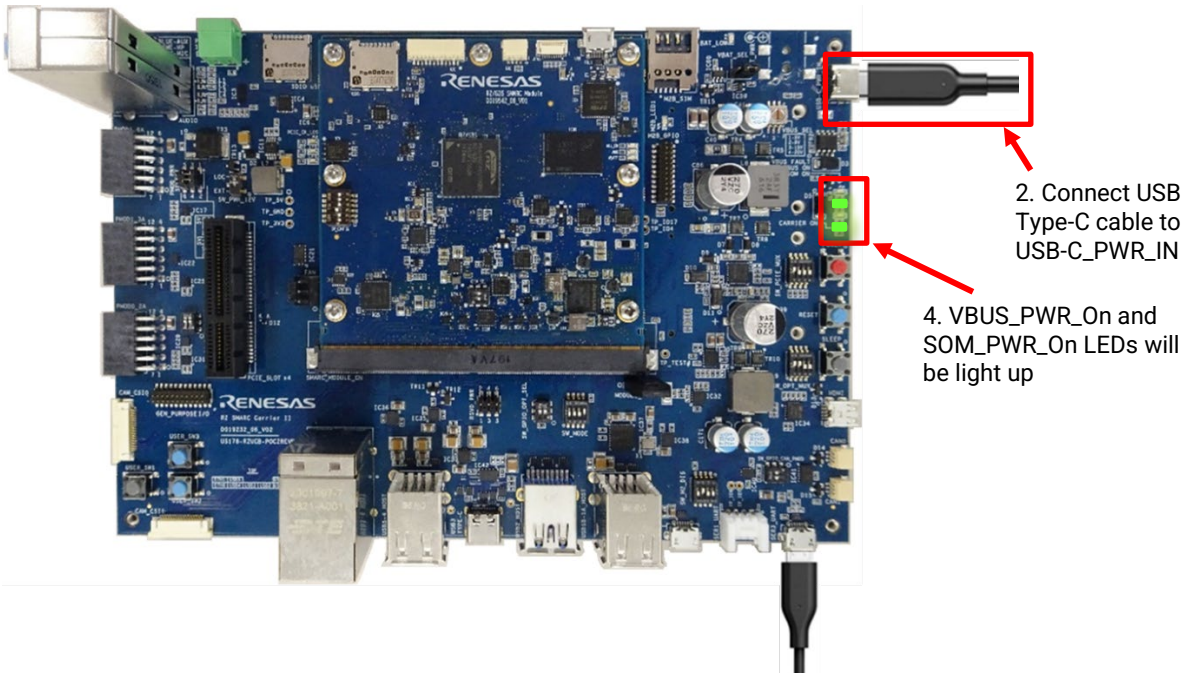
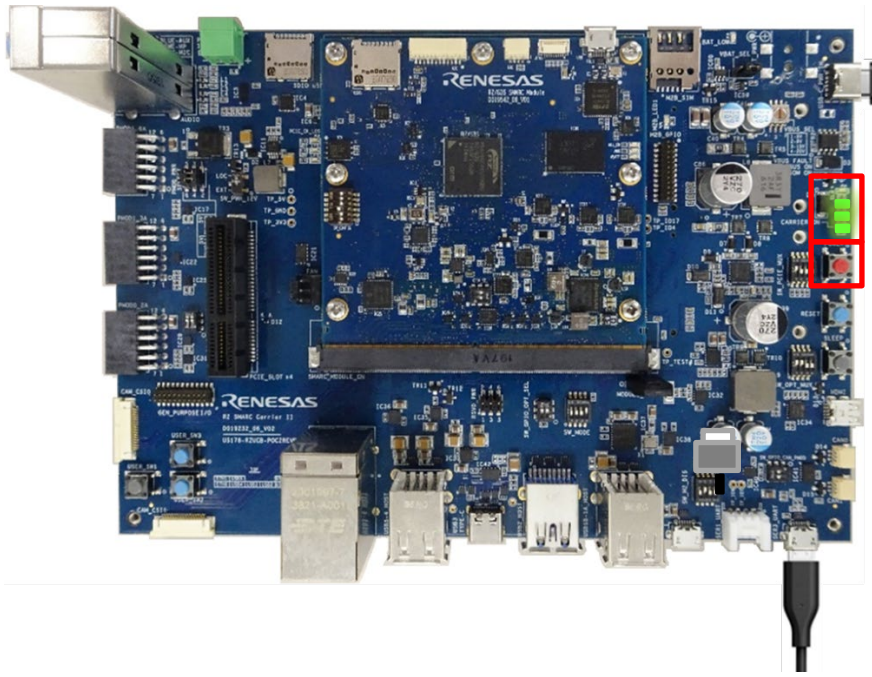


Figure 11. Connecting Power Supply

5. Push the “Power” red button for 1 msec 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 12. Power ON

4.2.2 Building files to write

The evaluation boards use the files in the Table 11 as the boot loaders. The boot loaders files are generated by following the build instruction. Please refer Table 3. Once builded copy these files to a PC which runs serial terminal software.

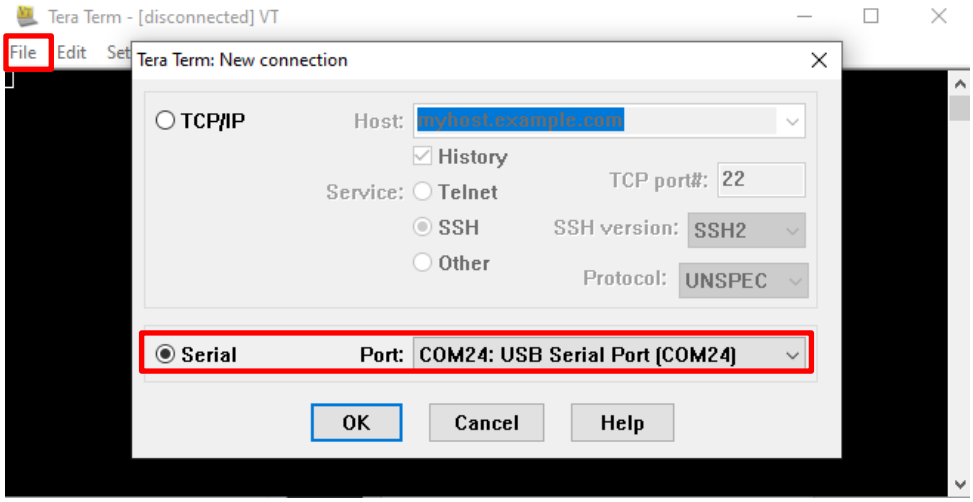
Table 10. File names of Boot loader

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

4.2.3 Settings

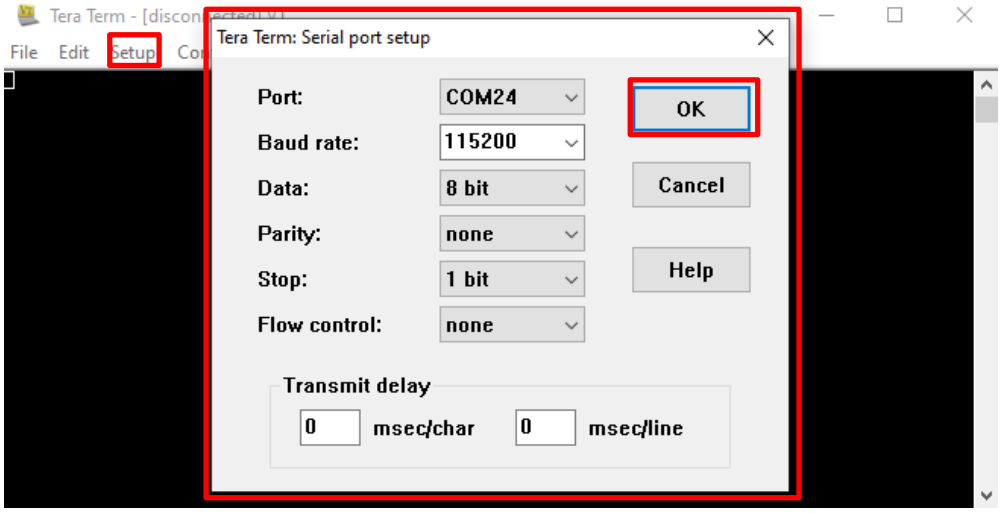
Connect the board to a host PC using the USB serial cable according to the Release Note.

1. To set the board to SCIF download mode, set “SW_MODE" according to the switch setting in **SCIF Download Mode** of SW_MODE.
2. Power the G3S SMARC EVK according to Power On 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.



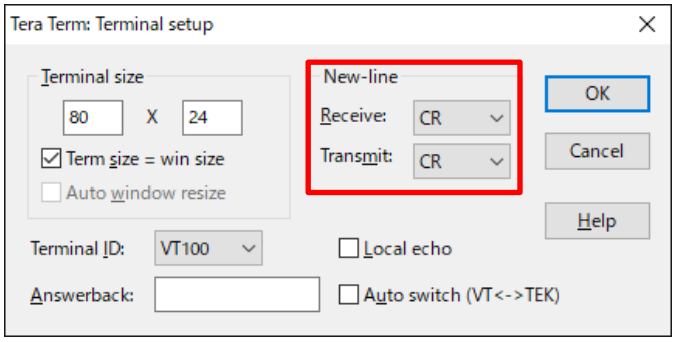
Set the settings about serial communication protocol on a terminal software as below:

- 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:

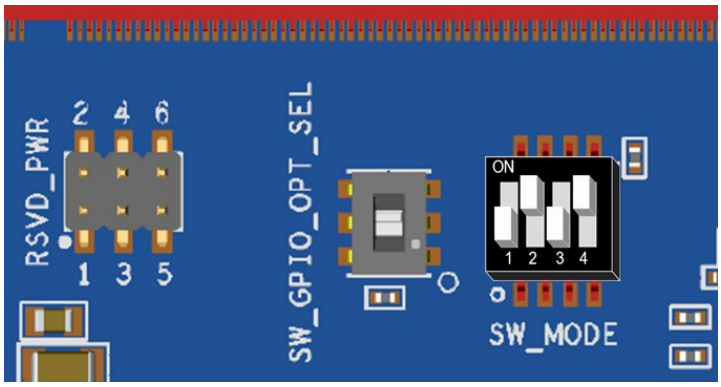


Table 11. SW_MODE

1	2	3	4
OFF	ON	OFF	ON

(2) After finished all settings, when pressed the RESET button, the messages below are displayed on the terminal.

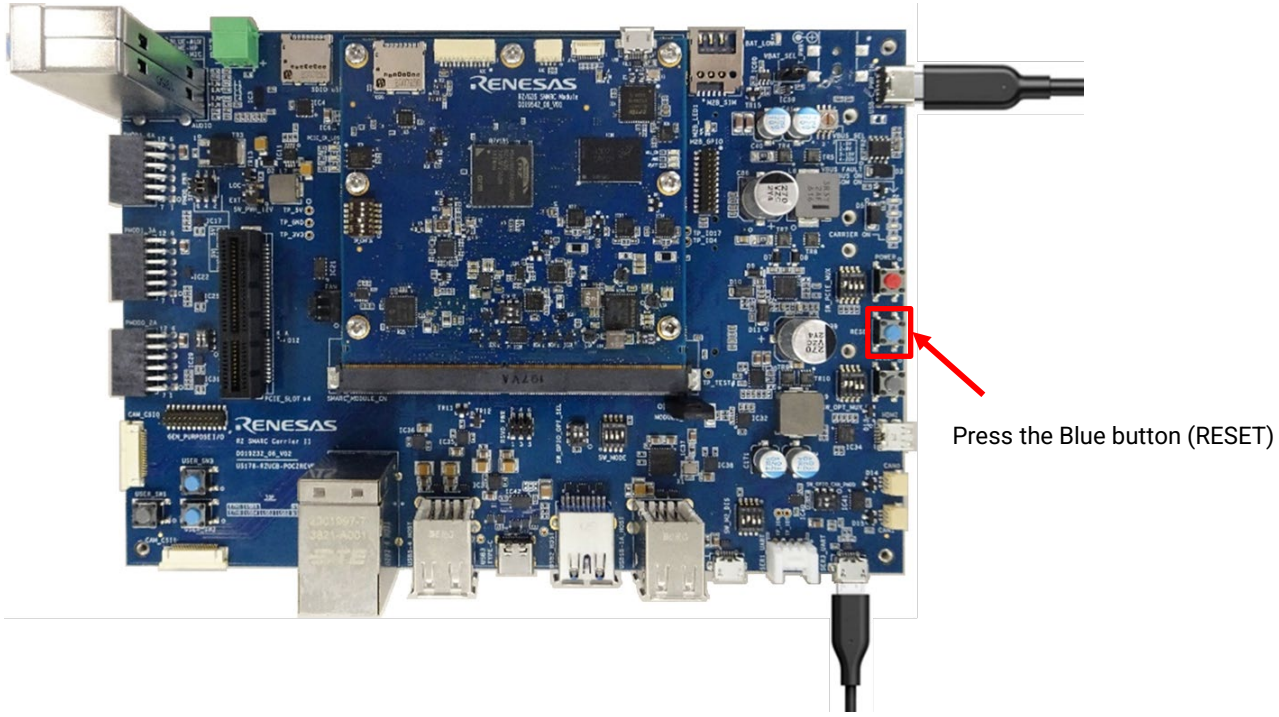


Figure 13. Press the RESET button

4.3 Download Flash Writer to RAM

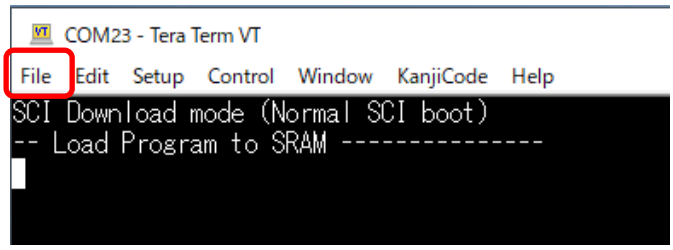
Turn on the power of the board by pressing PWR button. The messages below are shown on the terminal.

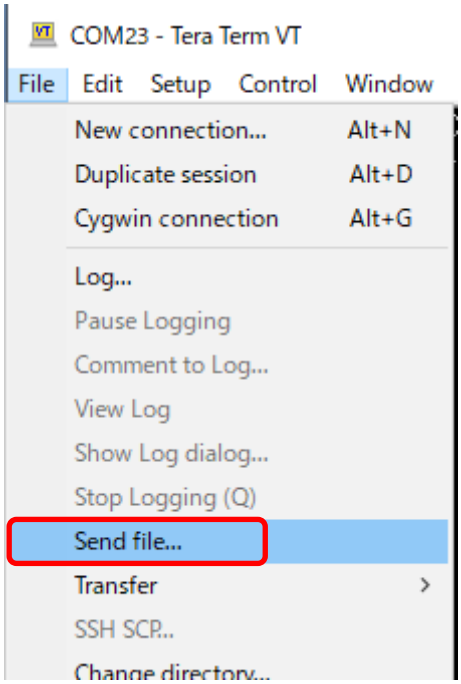
```
 SCIF Download mode
 (C) Renesas Electronics Corp.

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

Send an image of Flash Writer (FlashWriter-smarc-rzg3s.mot) using terminal software after the message “please send !” is shown.

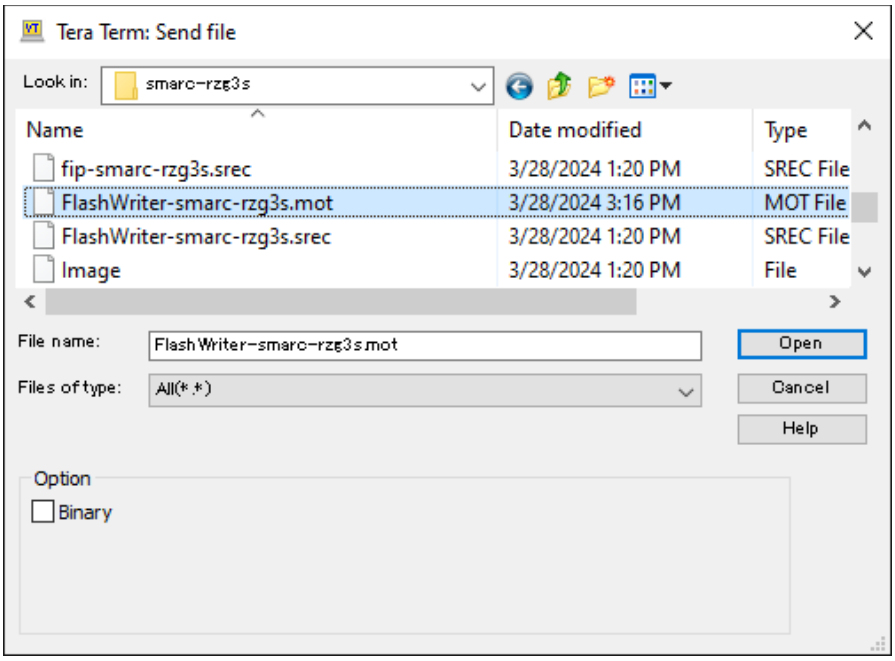
Below is a sample procedure with Tera Term.

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

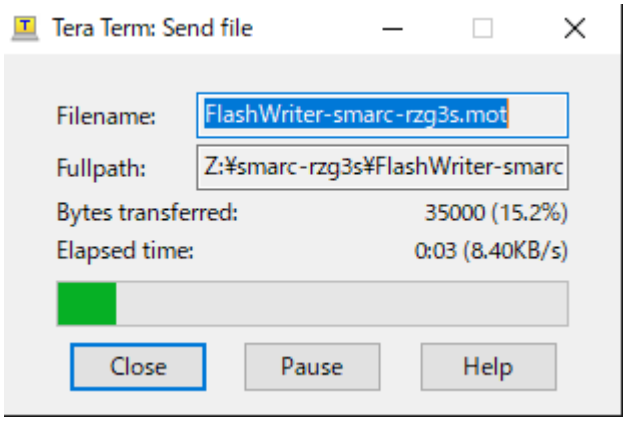




Then, select the image to be send and click “Open” button.



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
 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 are depending on each target board as described in Table 13.

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/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 download 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'64000
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 download 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'00064000
SpiFlashMemory End Address : H'0014782E
============================================================
```

```
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.

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

Table 12. 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	64000

4.5 Change Back to Normal Boot Mode

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

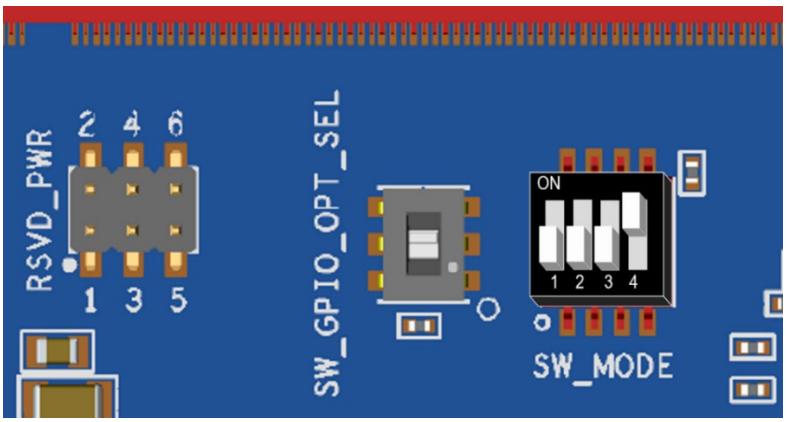


Table 13. SW_MODE

1	2	3	4
OFF	OFF	OFF	ON

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

```
U-Boot 2021.10 (Oct 25 2023 - 08:58:16 +0000)

CPU:   Renesas Electronics K rev 11.2
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.

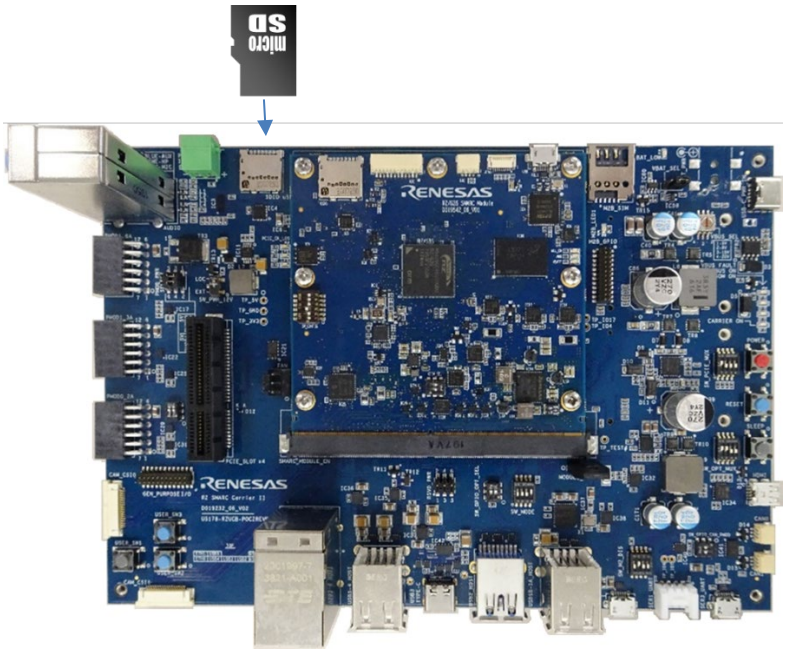


Figure 14. Set micro SD card to SMARC-EVK

Now the board can bootup 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 BSP/VLP are written to your board.

```
U-Boot 2021.10 (Oct 25 2023 - 08:58:16 +0000)

CPU:   Renesas Electronics K rev 11.2
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 ...

[    0.000000] Booting Linux on physical CPU 0x0000000000 [0x412fd050]
[    0.000000] Linux version 5.10.145-cip17-yocto-standard (oe-user@oe-host) (aarch64-
poky-linux-gcc (GCC) 8.3.0, GNU ld (GNU Binutils) 2.31.1) #1 SMP PREEMPT Sat Feb 27 0
2:21:18 UTC 2021
[    0.000000] Machine model: Renesas SMARC EVK version2 based on r9a08g045s33 SoC
 :
 :
 :
[    3.590799] systemd[1]: Detected architecture arm64.

Welcome to Poky (Yocto Project Reference Distro) 3.1.21 (dunfell)!

[    3.626305] systemd[1]: Set hostname to <smarc-rzg3s>.
[    3.636147] random: systemd: uninitialized urandom read (16 bytes read)
[    3.642972] systemd[1]: Initializing machine ID from random generator.
[    3.649771] systemd[1]: Installed transient /etc/machine-id file.
[    3.968222] systemd[1]: Configuration file /lib/systemd/system/watchdog.service is
marked executable. Please remove executable permission bits. Proceeding anyway.
[    4.193723] random: systemd: uninitialized urandom read (16 bytes read)
[    4.200871] systemd[1]: system-getty.slice: unit configures an IP firewall, but the
 local system does not support BPF/cgroup firewalling.
[    4.213244] systemd[1]: (This warning is only shown for the first unit using IP fir
ewalling.)
[    4.225485] systemd[1]: Created slice system-getty.slice.
[  OK  ] Created slice system-getty.slice.
 :
 :
 :
         Starting Update UTMP about System Runlevel Changes...
[  OK  ] Started Update UTMP about System Runlevel Changes.
[  OK  ] Started Hostname Service.

Poky (Yocto Project Reference Distro) 3.1.21 smarc-rzg3s ttySC0

BSP: RZG3S/RZG3S-SMARC-EVK/1.0.0
LSI: RZG3S
Version: 1.0.0
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, confirm that CARRIER_PWR_ON LED is off.

6. Building the SDK

To build Software Development Kit (SDK), run the commands below after the steps (1) – (3) of the section 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
```

Or

```
$ cd ~/rzg3s_vlp_<package version>/build
$ MACHINE=smarc-rzg3s bitbake core-image-bsp -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-bsp-aarch64-smarc-rzg3s-toolchain-<version>.sh
```

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

```
$ cd ~/rzg3s_vlp_<package version>/build
$ MACHINE=smarc-rzg3s bitbake core-image-bsp -c cleanall
$ MACHINE=smarc-rzg3s bitbake core-image-bsp
```

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 BSP. 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-bsp-aarch64-smarc-rzg3s-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:

```
$ sudo sh ./rzg3s_vlp_<package version>/build/tmp/deploy/sdk/poky-glibc-x86_64-core-im
age-bsp-aarch64-smarc-rzg3s-toolchain-<version>.sh
Poky (Yocto Project Reference Distro) SDK installer version 3.1.31
==================================================================
Enter target directory for SDK (default: /opt/poky/3.1.31): ~/workspace/rzg3s/bspf304-
build-test/build/tmp/deploy/sdk/temp
You are about to install the SDK to "/home/renesas/workspace/rzg3s/bspf304-build-test/
build/tmp/deploy/sdk/temp". 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/3.1.31/environment-setup-aarch64-poky-linux
```

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/3.1.31/environment-setup-aarch64-poky-linux
```

7.1.2 How to build Linux application

Step 1. Make a work 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):
  $(CC) -o $(APP) $(SRC) $(CFLAGS)

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

clean:
  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/3.1.31/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.

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.

```
BSP: RZG3S/RZG3S-SMARC-EVK/<package version>
LSI: RZG3S
Version: <package version>
smarc-rzg3s login: root
root@smarc-rzg3s:~# /usr/bin/linux-helloworld

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

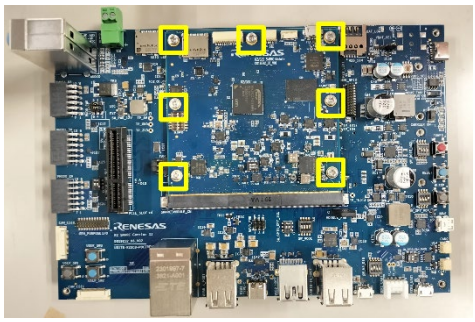
Note: Refer to the start-up guide for the method of how to boot the board and system.

8. Appendix

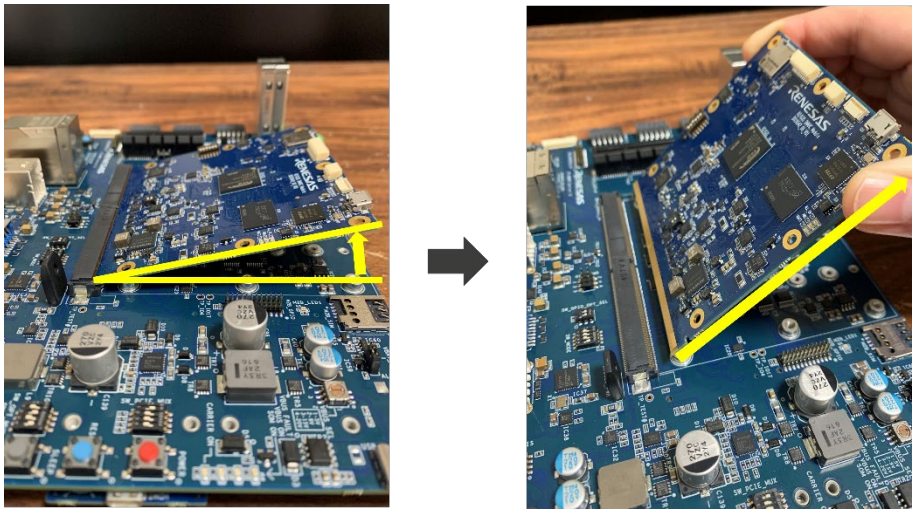
8.1 How to replace the SMARC Module Board

Please be careful when replacing the board as follows.

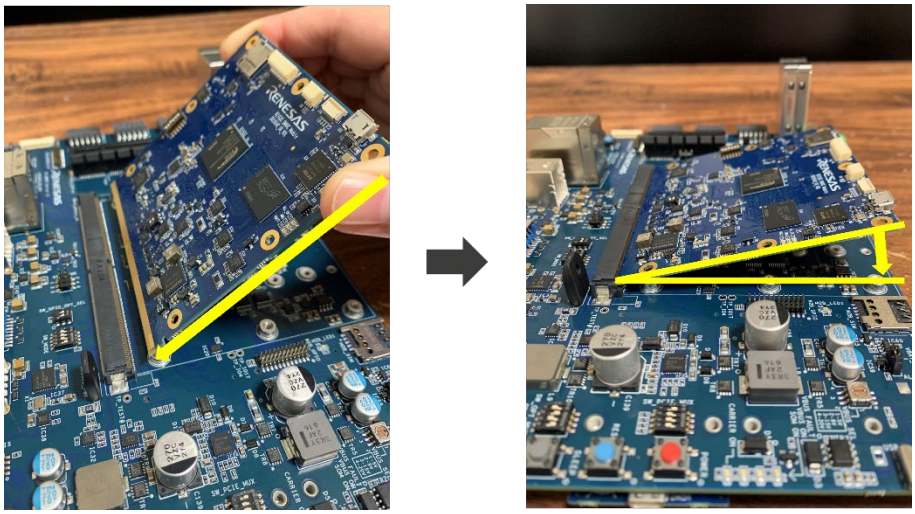
1. Remove the four screws.



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 is described.

8.2.1 Writing Bootloader for eMMC Boot

For the boot operation, EXT_CSD register of eMMC need 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 4.3Download Flash Writer to RAM),

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_emmc-smarc-rzg3s.srec” from terminal software after the message “please send !” is shown.

After successfully download 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 :320
```

```
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 download the binary, messages like below are shown on the terminal.

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

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
>
```

Table 14. Address of EXT_CSD register of eMMC for eMMC boot

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

Table 15. 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_emmc-smarc-rzg3s.srec	1	00000001	a1E00
fip-smarc-rzg3s.srec	1	00000320	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

Create a microSD card by 3.2 Write files to the microSD card (not used wic image). After finishing, please 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/smarc-rzg3s/<root filesystem> ./
$ cd $WORK
$ sudo umount /media/user/rootfs
```

The file nemes of *<root filesystem>* are listed in the Table 3.

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:

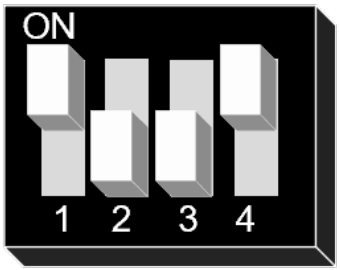


Table 16. "SW_MODE"

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 power of 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/mmcblk0p2 /mnt/
root@smarc-rzg3s:~# tar xf /home/root/core-image-bsp-smarc-rzg3s.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 2021.10 (Oct 25 2023 - 08:58:16 +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 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 the step (2) in the section 2, run the following commands. Once executed, proceed until step (5) in the section 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 the 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-rzg3s:~# systemctl status docker.service
root@smarc-rzg3s:~# docker version
```

(4) 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!
```

8.4 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                                                |
| 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 4.4Write the Bootloader. Upload with the “ascii” command in the same way as a Flash Writer file.

8.5 Device drivers

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

File	Description
RTK0EF0045Z900xxxxAZJ-v3.0.x.zip	BSP Manual Set for RZ/G3S.

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

Table 17. Support device drivers

Device Driver	Documents
Kerne l Core	r01us0623ej0xxx-rz-g_Kernel_Core_UME.pdf
GPIO	r01us0624ej0xxx-rz-g_GPIO_UME.pdf
Direct Memory Access Controller (DMAC)	r01us0625ej0xxx-rz-g_DMAC_UME.pdf
Multi-function Timer Unit 3a (MTU3a)	r01us0626ej0xxx-rz-g_MTU3a_UME.pdf
Port Output Enable 3 (POE3)	r01us0627ej0xxx-rz-g_POE3_UME.pdf
General PWM Timer (GPT)	r01us0628ej0xxx-rz-g_GPT_UME.pdf
Watchdog Timer (WDT)	r01us0629ej0xxx-rz-g_WDT_UME.pdf
Serial Communication Interface with FIFO A (SCIFA)	r01us0630ej0xxx-rz-g_SCIFA_UME.pdf
I2C Bus Interface	r01us0631ej0xxx-rz-g_I2C_UME.pdf
I3C Bus Interface	r01us0632ej0xxx-rz-g_I3C_UME.pdf
Renesas Serial Peripheral Interface (RSPI)	r01us0633ej0xxx-rz-g_RSPI_UME.pdf
SPI Multi IO Bus Controller	r01us0634ej0xxx-rz-g_xSPI_UME.pdf
RS-CANFD Interface	r01us0635ej0xxx-rz-g_CANFD_UME.pdf
Gigabit Ethernet Interface	r01us0636ej0xxx-rz-g_Gigabit_Ethernet_UME.pdf
USB 2.0 Host	r01us0637ej0xxx-rz-g_USB2.0_Host_UME.pdf
USB 2.0 Function	r01us0638ej0xxx-rz-g_USB2.0_Function_UME.pdf
SD/MMC Host Interface	r01us0639ej0xxx-rz-g_SD_MMC_UME.pdf
Serial Sound Interface	r01us0640ej0xxx-rz-g_SSIF_UME.pdf
A/D Converter	r01us0641ej0xxx-rz-g_AD_Converter_UME.pdf
Thermal Sensor Unit (TSU)	r01us0642ej0xxx-rz-g_Thermal_Sensor_UME.pdf
Power Management	r01us0643ej0xxx-rz-g_Power_Management_UME.pdf
PCI Express Interface	r01us0646ej0xxx-rz-g_PCIe_UME.pdf
Renesas SPDIF Interface	r01us0680ej0xxx-rz-g_SPDIF_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	Jun. 30, 2023	-	First edition issued.
1.01	Nov 30, 2023	38	Add eMMC boot process.
1.02	Apr 24, 2024	6	Add “2.1 Required Host OS” section.
		43	Add “8.3 Docker” section.
1.03	May 31, 2024	-	VLP/G v3.0.6-update1
		45	Add “8.4 Booting Setup with Ubuntu PC” section.
		47	Add “8.5 Device drivers” section.

Website and Support

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

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