AVNET
Reach Further™

/ RZBOARD V2L

Linux Yocto User Manual

v2.1

Copyright Statement:

Disclaimer:

Revision History

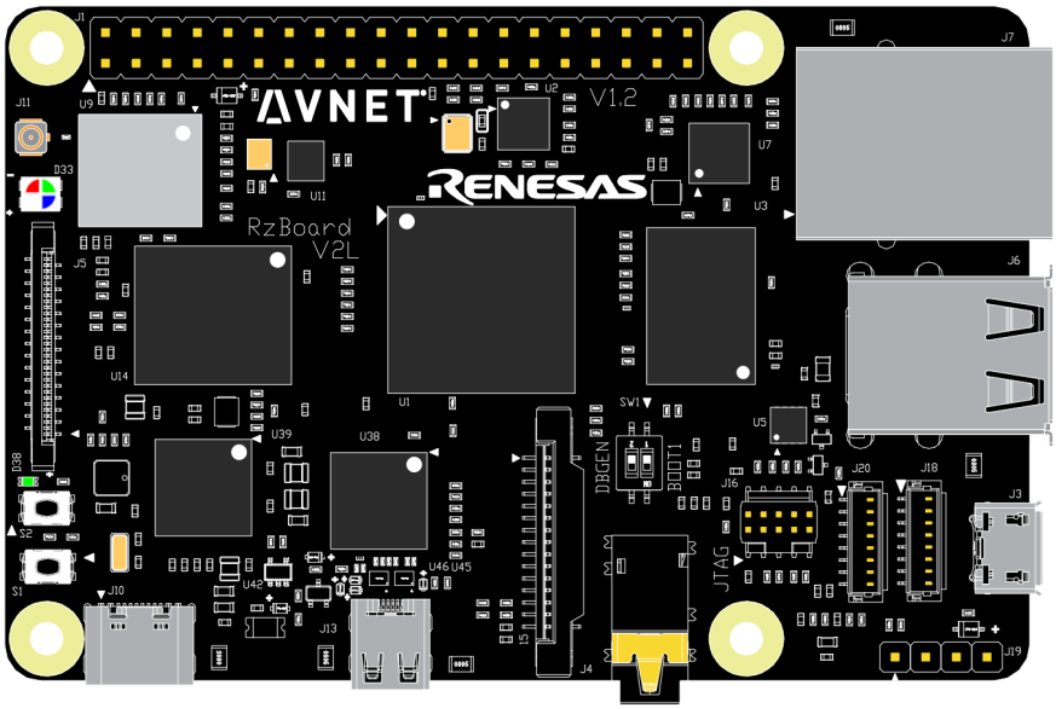
Rev.	Description	Author	Date
v1.0	Initial version	Lily	2022/9/21
v2.0	Updated Yocto Project to 3.1.14	Lily	2022/09/28
v2.1	Edits to all sections (eliminated non-scripted flash writes)	Peter	2022/10/26

Contents

Chapter 1 Introduction

1.1 Target Board

RzBoard V2L is a development board developed by Avnet, based on the RZ/V2L group of 64bit Arm-based MPUs from Renesas Electronics.

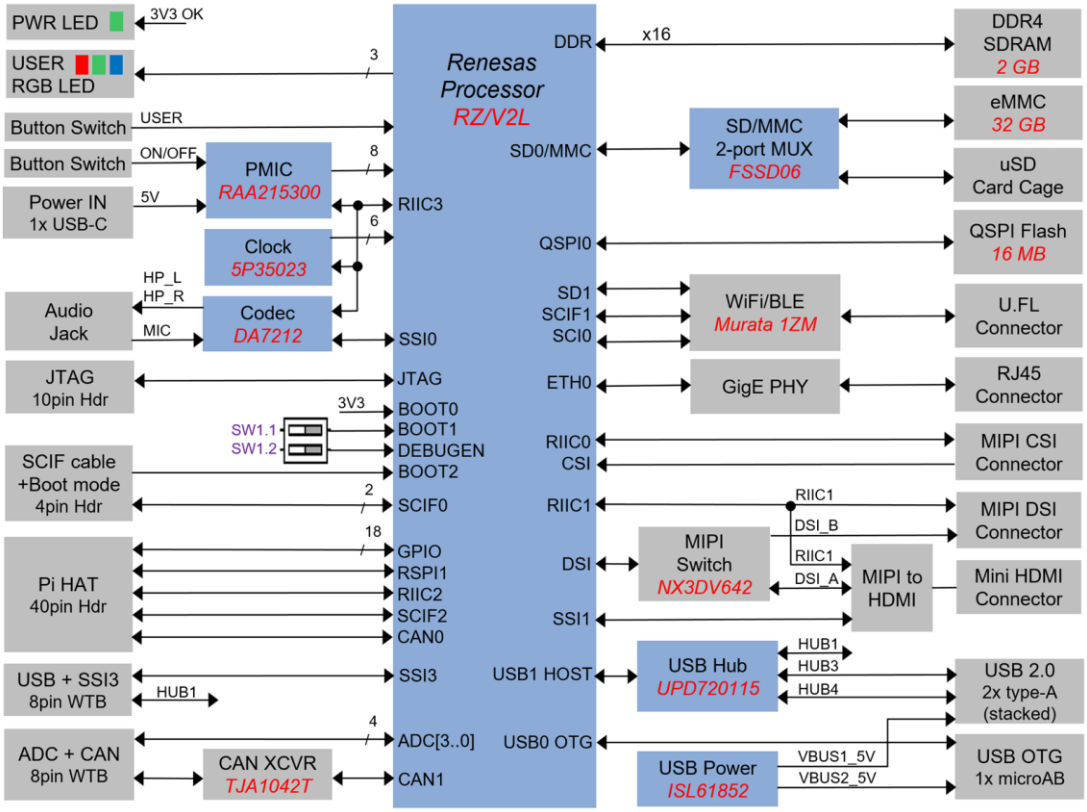


1.2 Introduction

This document provides a guide to prepare RzBoard to boot up with the Verified Linux Package for RZ/V2L Group and introduces how to use the supported RZBoard functions.

1.3 Feature List

- Yocto version: Dunfell (3.1.14)
- U-Boot version: 2021.10
- Kernel version: 5.10.83
- Evaluation image: Yocto Image
- QSPI boot / eMMC boot
- Device-tree overlay support
- Desktop (Weston 8.0)
- 1 Gigabit Ethernet (RJ45)
- 2 x USB 2.0 Host + 1 x USB 2.0 OTG
- 2 UART (TTL) include debug port
- External interfaces (I2C, UART, SPI, CANFD and GPIO)
- 802.11ac Wi-Fi plus BLE 5.0
- MIPI-DSI display
- Audio playback and record
- MIPI-CSI Camera / USB Camera
- DRP-AI support
- H.264 hardware Enc/Dec support

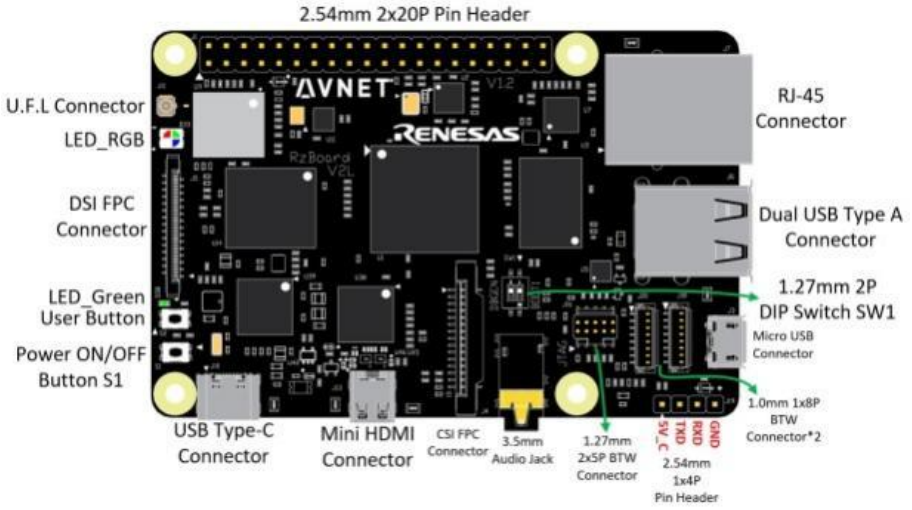


Chapter 2 System Boot-Up

2.1 Preparation and Programming

2.1.1 Hardware Preparation

- USB power source with USB type-C cable
- USB to Serial adapter cable
- Ethernet cable
- Connect J19 p2-p4 (TXD,RXD,GND) to the USB to Serial cable, then connect this cable to the PC.



2.1.2 Software Source Files Preparation

The RzBoard uses the files in the Table 2 as bootloader and system images. These can be rebuily using the procedure described in ***RzBoard-Linux-Yocto-Lite-Development_Guide-V*.*-EN.pdf***

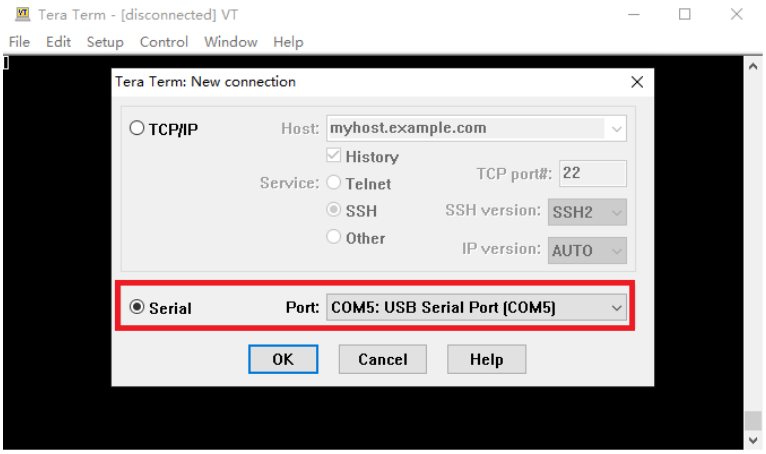
Files	File Name	Description
Flash Writer	*flashwriter_rzboard.mot*	FlashWriter image tool, which is used to flash bootloader images into QSPI or eMMC, can be downloaded from the Host PC via SCIF by boot ROM
Bootloader	*bl2_bp-rzboard.srec*	Bootloader image in Motorola S-Record format, ARM TFA(Trusted Firmware-A) BL2 image
	fip-rzboard.srec	Bootloader image, ARM TFA(Trusted Firmware-A) BL31 and u-boot combined image
System Image	*core-image-rzboard-********** **********.rootfs.wic*	system image, include linux kernel, DTB and root filesystem, Need to be written to TF card or eMMC

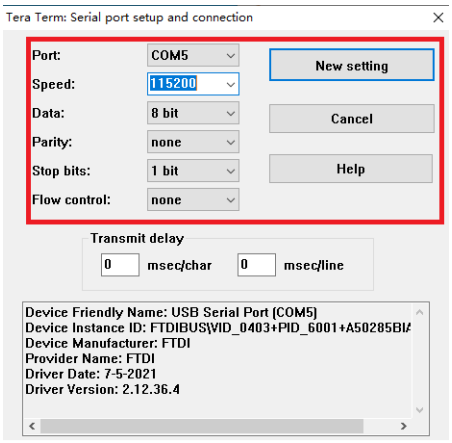
Table 1: Image Files

2.1.3 Software Tools Preparation

Install Tera Term terminal software

- For Windows-based write of bootloader images, command-line debug output and command entry, the use of ***Tera Term*** terminal software is recommended
- Download and install ***teraterm-***.exe*** and configure the relevant COM port as shown below:





Install Fastboot

Download ***Fastboot*** (Windows version) tools from the Android Platform Tools official website.

2.1.4 Procedure to Reflash the Bootloader Firmware (eMMC)

.BAT File Name	File Size	File Names	Boot Mode Board Settings
flash_bootloader.bat Download Type: SCIF0 *@115.2 kb/s*	268 KB 115 KB 2.02 MB	*flashwriter_RZBoard.mot* *bl2_bp-RZBoard.srec* *fip-RZBoard.srec*	BOOT2=1: Fit fly-wire from J1 pin2 to J19 pin1 BOOT1=0: Set SW1.1 = ON BOOT0=1: Remove SD card

(Programming bootloader images is less frequently required than updates to the Linux System Image)

Related tools, scripts and relevant information are kept up to date in the following repo:
https://github.com/Avnet/rzboard-program-tools

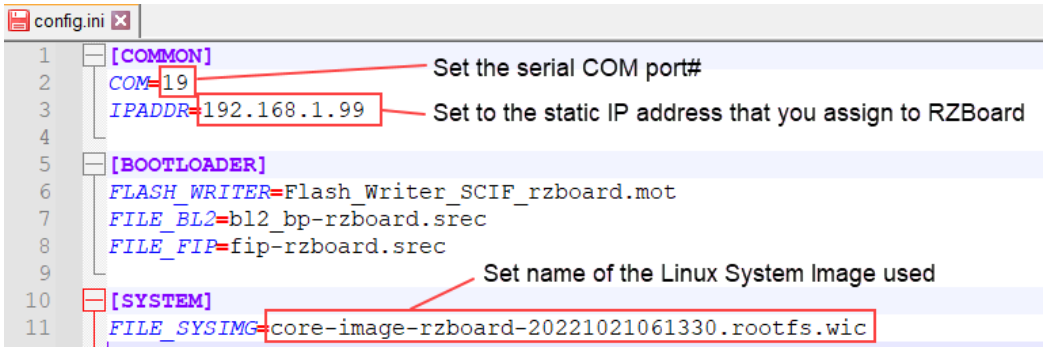
Simply Download latest image files, .bat and macro files using the following URL:
https://avnet.me/RZBoard_emmc

An easy scripted procedure is provided to program the following pre-built bootloader image files via the SCIF interface (ie. USB-Serial cable) into QSPI or eMMC flash memory on RZBoard:

- **flashwriter_RZBoard.mot** FlashWriter image tool
  Once downloaded, this is used to program the following two bootloader images into eMMC
- **bl2_bp-RZBoard.srec** bootloader image in Motorola S-Record format, ARM TFA (Trusted Firmware-A) BL2 image
- **fip-RZBoard.srec which** is a combination of bootloader image, ARM TFA (Trusted Firmware-A) BL31 and u-boot combined image

Note: Complete steps 1-6 below, prior to running the provided ***flash_bootloader.bat*** file (step 7)

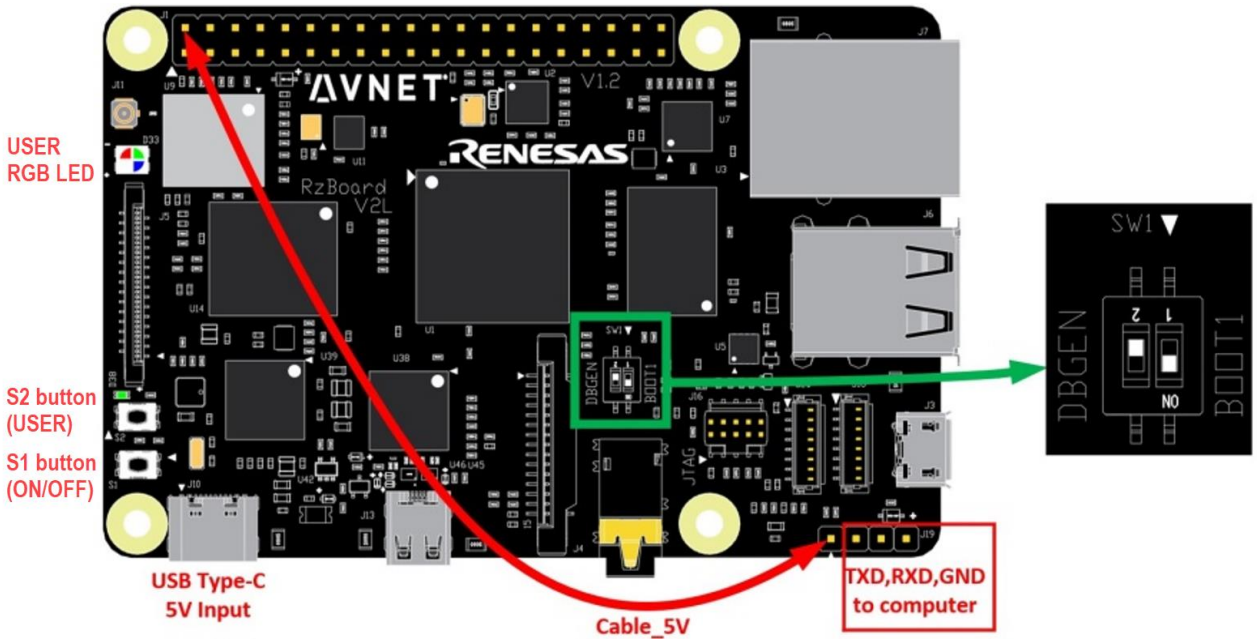
1) Download the latest image files, .bat and macro files from https://avnet.me/RZBoard_emmc and extract the zipped files to a staging folder on the development computer
2) Edit **Windows** Ethernet network adapter settings for the development computer:
   Set it's **IPv4** properties to static IP Address **192.168.1.88**
3) In the staging folder, edit the **config.ini** file (update the COM port#, the IP Address and ensure that this lists the matching filenames names for **Bootloader** image files and the Linux **System** image file)



4) Power-off RZBoard

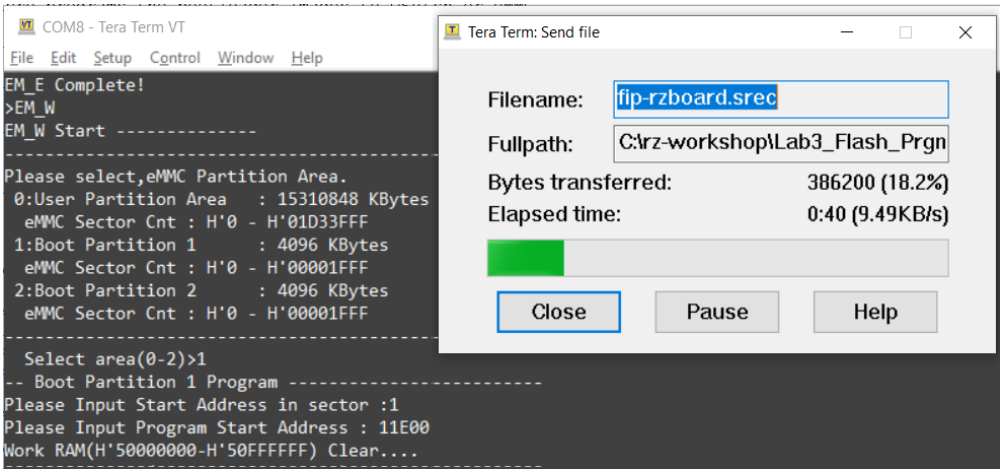
5) Place RZBoard into "**SCIF download boot-mode**" by setting **BOOT[2:0]** to **b101** ie.
   - Set **BOOT2**=**1** by strapping **J19-pin1** to **+5V** (ie. connect it to **J1-pin2** on the 40pin header)
   - Set **BOOT1**=**0** by strapping **SW1.1 = ON**
   - Set **BOOT0**=**1** by **removing SD card** from MicroSD slot

6) On RZBoard's **J19** *Debug UART* 4-pin header, connect the fly-leads from the USB-Serial cable connected to the development computer.



7) Run ***flash_bootloader.bat*** (to launch Tera Term macro using the edited ***config.ini*** settings)
Choose the media (eMMC or QSPI Flash) to program, the macro then waits for system power up.

8) Press and hold **S1** for 2 seconds to power-on RZBoard,
the macro will now proceed. Wait for this to complete (<5 min)



2.1.5 Procedure to Reflash the Linux System Image (eMMC)

.BAT File Name	File Size	File Names	Boot Mode Settings
flash_system_image.bat Download Type: Ethernet Fastboot *@1 Gb/s*	2.53 GB *typical	***core-image-RZBoard-2022 0920085823.rootfs.wic***	BOOT2=0: **Remove fly-wire** from **J1** pin2 to **J19** pin1 BOOT1=0: Set **SW1.1 = ON** BOOT0=1: **Remove SD card**

A scripted procedure is provided to program the large Linux System Image file, into RZBoard's eMMC flash memory, via Gigabit Ethernet from the development computer.
Note: Complete steps 1-6 below, prior to running the provided ***flash_system_image.bat*** file:

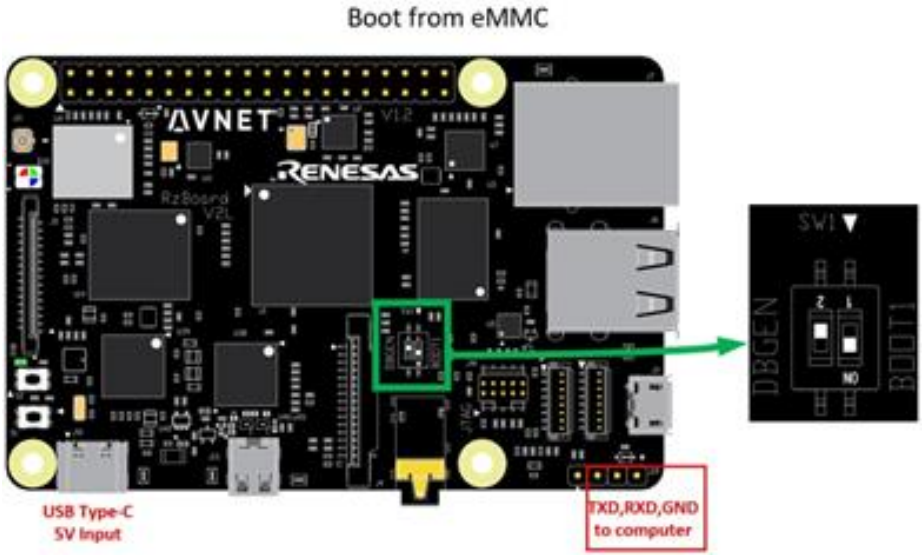
1) Download the image files, .bat and macro files from https://avnet.me/RZBoard_emmc and extract the zipped files to a staging folder on the development computer

2) Edit **Windows** Ethernet network adapter settings for the development computer:
Set it's **IPv4** properties to static IP Address **192.168.1.88**

3) Edit the **config.ini** file (update the COM port#, the IP address and name of the System image file)

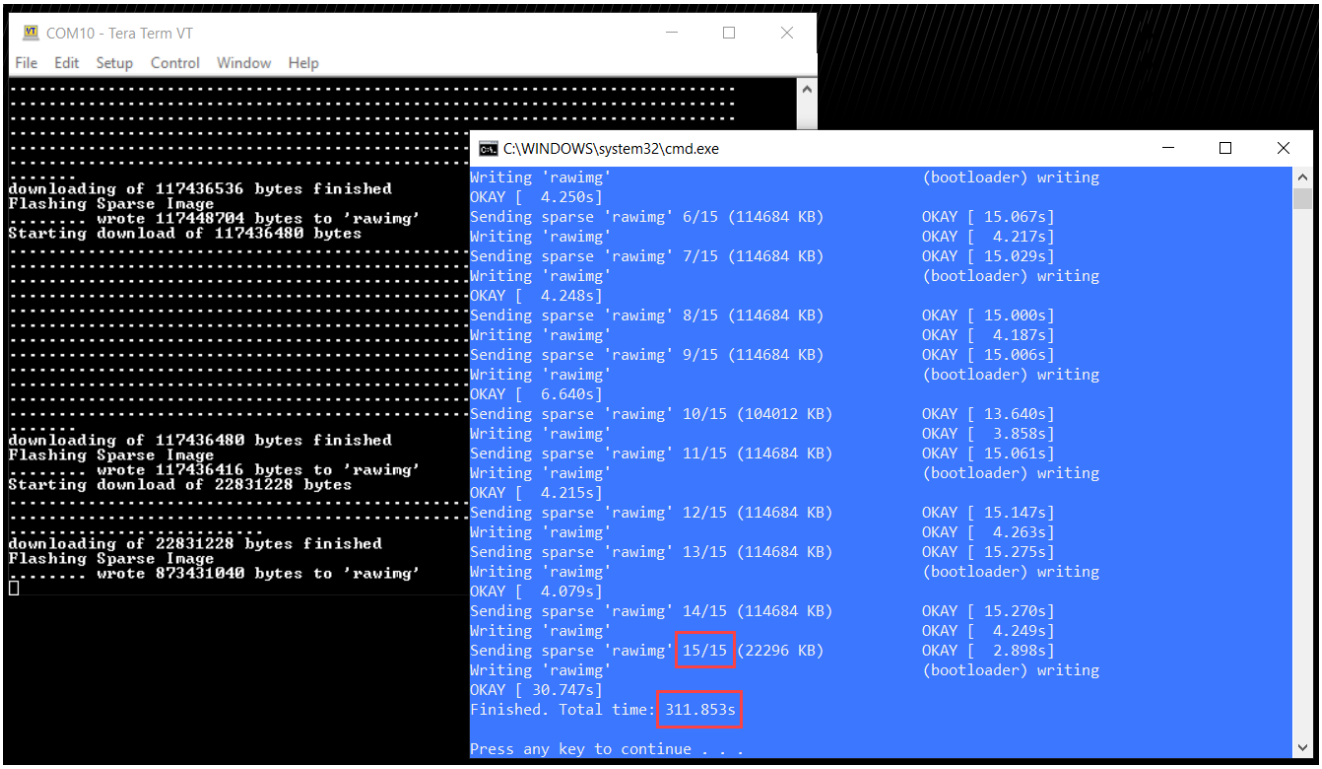
4) Power-off RZBoard

5) Place RZBoard into "**eMMC (1V8) boot-mode**" by setting **BOOT[2:0]** to **b001** (as tabled above), ie.
   - Set **BOOT2**=**0** by removing fly-wire from **J19-pin1** to **J1-pin2** (40pin header)
   - Set **BOOT1**=**0** by strapping **SW1.1 = ON**
   - Set **BOOT0**=**1** by **removing SD card** from MicroSD slot



6) Run ***flash_system_image.bat*** (launches Tera Term macro using saved **config.ini** settings)

7) Power-on RZBoard. Ethernet connetion will be established and a blue window shall open in <30 sec.



8) Wait for the macro to complete (typically 15 blocks of data get sent and this completes in <5 min). No input or operation is required during this period. After finishing, **press any key** to exit the BAT script.

9) Now set RzBoard to boot from QSPI or eMMC as needed and power-cycle the board using switch S1.

2.2 Booting RZBoard

RzBoard supports Linux boot from eMMC ***or*** SD card.
Before attempting to boot Linux system image from eMMC make sure the SDCard is ***not*** in the slot.

Two different methods of booting RZBoard are described in this section:

a) Booting u-boot and the Linux System image from eMMC

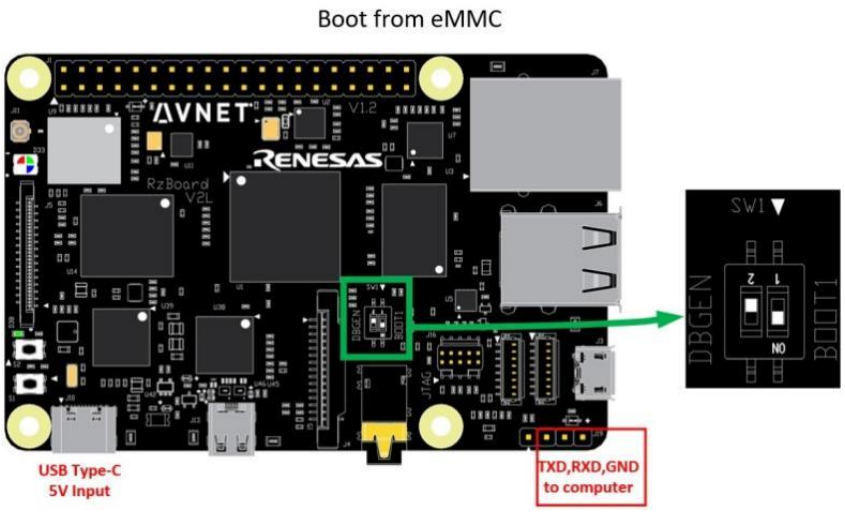
b) Booting u-boot from QSPI flash, booting Linux system image from SD card

For development, booting u-boot from QSPI flash, then using NFS (network file system) located on the development PC (via network Ethernet connection) will be detailed in a later version of this document.

2.2.1 Boot from eMMC

The least complex method, where u-boot and the Linux system image are booted from eMMC memory. After writing bootloader and linux system images into eMMC, boot RzBoard from eMMC as follows:

- Connect Boot2 (Pin1 of J19) to GND, Dial out SD card, Set SW1 as shown below:



- Connect suitable 5V power source to RZBoard via the **J10** USB type-C connector.
- Press and hold Power button **S1** for 2 seconds to power-on the system.
- When the system boots-up, the serial terminal will print the following information:

```
Welcome to rzboard Board GNU/Linux yocto(dunfell) system.
rzboard login:
```

- Enter username as “**root**”, password as “**avnet**” to login.
- The Linux system interface also supports directly attached keyboard and mouse

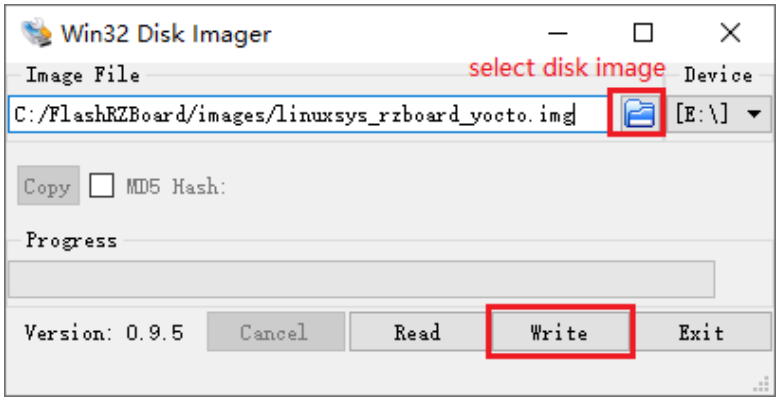
2.2.2 Boot using QSPI flash and SD Card

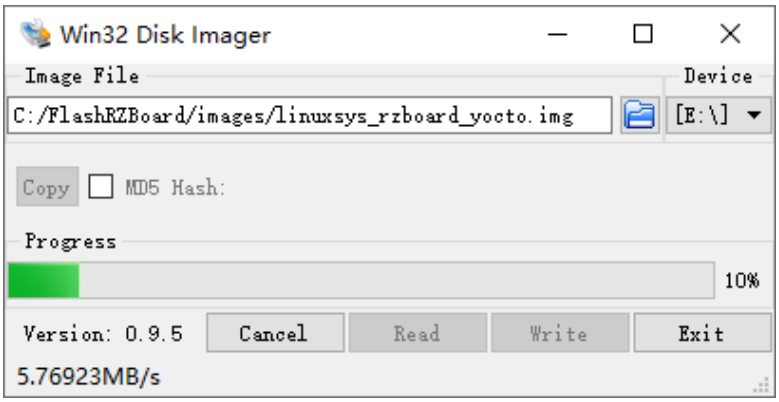
Booting u-boot from QSPI flash is typically when the Linux System image is on SD card or when NFS (network file system) is used. To boot RzBoard from QSPI flash, the two bootloader images (***bl2_bp-rzboard.srec*** and ***fip-rzboard.srec***) need to have been written into it, using the scripted Flash Writer procedure (as described earlier in this chapter)

2.2.3 How to Program Linux System Image into SDCard

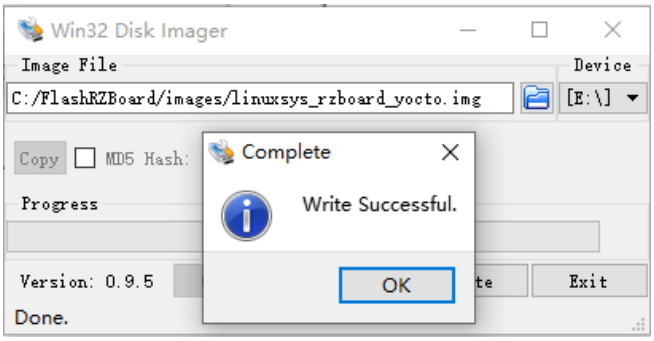
Under Windows OS, **Win32 Disk Imager** tool is used to write Linux system image into the SDcard.

- Insert the SD card into the card reader, then connect the card reader to the USB port on the PC.
- Open **Win32 Disk Imager** on the PC, Select the Disk Image, then click “Write”.



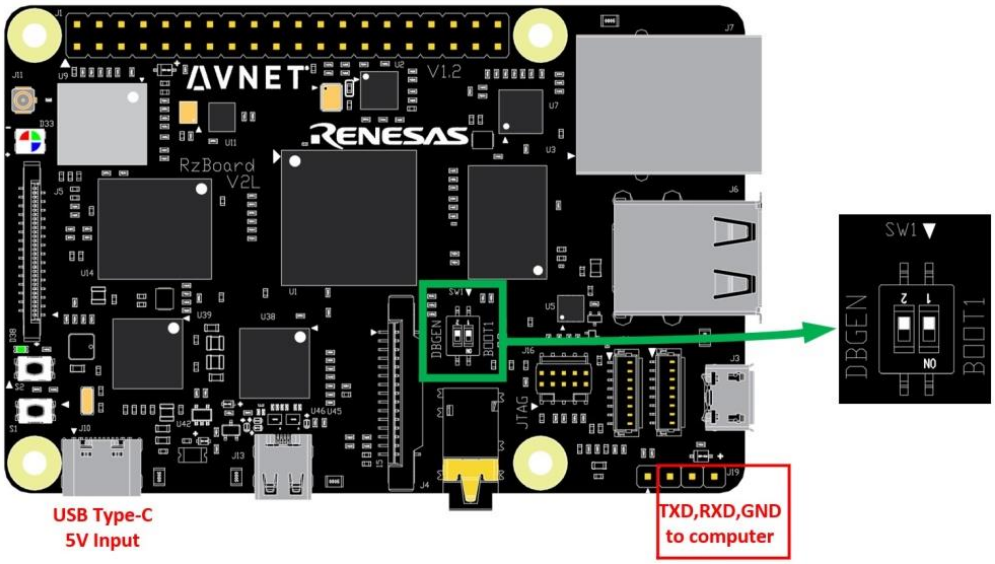


- Wait for completion of the write operation…



2.2.4 Boot up from QSPI

- Connect Boot2 of J19 to GND, Insert the SDcard, Set SW1 as shown below:



- Power on the board with a 5V, 2A, Type-C interface power (to J10).
- Press Power button **S1** for 2 seconds and system will power on.
- When the system boot up, the serial terminal will print the following information:

```
Poky (Yocto Project Reference Distro) 3.1.14 rzboard ttySC0
rzboard login:
```

- Enter username as “**root**”, password as “**avnet**” to login.
- Users can also use keyboard and mouse connected to RzBoard to login to Linux.

Chapter 3 Feature Configuration & Introduction

In this chapter, we mainly introduce the features of RzBoard. First of all, please refer to *RzBoard-Start-up-Guide-V*.*.pdf* and boot up the system refer to the previous chapter. Configure or use the functions according to the following guidance.

3.1 Settings in uEnv.txt

User could configure some environment variables in uEnv.txt, which can be loaded in the U-boot stage. The uEnv.txt file has a very simple file format. The format is a single *property=value* statement on each line, where value is either an integer or a string. Comments may be added, or existing config values may be commented out and disabled, by starting a line with the # character.

The device-tree overlay function is supported from this version and the device-tree overlay file (*.dtbo) is placed in the overlay/ directory in the FAT partition of the SDIO card or eMMC. To load the device-tree overlay file (*.dtbo), you need to set "*fdt_extra_overlays*" and "*enable_overlay_*" prefix variable in uEnv.txt. You can also add other configurations defined in u-boot to the uEnv.txt file.

The specific description is as follows:

```
/-----------------------|--------------|-------------------------------
|       Config          | Value if set |     To be loading
|-----------------------|--------------|-------------------------------
| enable_overlay_disp   |  'hdmi'      |  rzboard-hdmi.dtbo
|                       |  'mipi'      |  rzboard-mipi.dtbo
|-----------------------|--------------|-------------------------------
| enable_overlay_camera |  'ov5640'    |  rzboard-ov5640.dtbo
|                       |  'as0260'    |  rzboard-as0260.dtbo
|-----------------------|--------------|-------------------------------
| enable_overlay_adc    | '1' or 'yes' |  rzboard-adc.dtbo
|-----------------------|--------------|-------------------------------
| enable_overlay_can    | '1' or 'yes' |  rzboard-can.dtbo
|-----------------------|--------------|-------------------------------
| enable_overlay_cm33   | '1' or 'yes' |  rzboard-cm33.dtbo
|-----------------------|--------------|-------------------------------
| enable_overlay_audio  | '1' or 'yes' |  rzboard-lite-audio.dtbo
|-----------------------|--------------|-------------------------------
| enable_overlay_i2c    | '1' or 'yes' |  rzboard-ext-i2c.dtbo
|-----------------------|--------------|-------------------------------
| enable_overlay_spi    | '1' or 'yes' |  rzboard-ext-spi.dtbo
|-----------------------|--------------|-------------------------------|
| enable_overlay_uart2  | '1' or 'yes' |  rzboard-ext-uart2.dtbo
|-------------------------------------------------------------------
| fdtfile              : is a base dtb file, should be set rzboard.dtb
|-------------------------------------------------------------------
| fdt_extra_overlays   : other dtbo files to be loading, such as rzboard-f1.dtbo rzboard-f2.dtbo
|-------------------------------------------------------------------
|-------------------------------------------------------------------
|  uboot env   : you could set some environment variables of u-boot here, such as 'console=' 'bootargs='
\-------------------------------------------------------------------
```

Note: *fdtfile* must be set to a device tree binary blob, which is the basis for applying dtbo file. *fdtfile* should be set, other configurations are optional.

Here is the default setting in uEnv.txt:

```
fdtfile=rzboard.dtb
enable_overlay_disp=hdmi
#fdt_extra_overlays=1.dtbo 2.dtbo 3.dtbo
                                    #ethaddr=aa:bb:cc:aa:bb:cc
```

Modify uEnv.txt methods:

We can find uEnv.txt in /boot, then use **nano** or **vi** command to edit the uEnv.txt.

```
root@rzboard:~# cd /boot
root@rzboard:/boot# ls
Image  cm33  overlays  readme.txt  rzboard.dtb  uEnv.txt
root@rzboard:/boot# vi uEnv.txt
```

We can edit the **uEnv.txt** as needed and save it.

```
# Refer to readme.txt for more information on setting up U-Boot Env

fdtfile=rzboard.dtb
enable_overlay_disp=hdmi
#fdt_extra_overlays=1.dtbo 2.dtbo 3.dtbo
#ethaddr=aa:bb:cc:aa:bb:cc
```

After the modification, execute *sync* and *reboot* command to make it effect.

3.2 User LED (RGB)

RzBoard has a tri-color RGB LED indicator available for user-defined functions.

It flashes blue to indicate heartbeat by default, but this LED can be controlled using follow commands:

LED output blue:

```
root@rzboard:~# echo default-on > /sys/class/leds/led_blue/trigger
root@rzboard:~# echo 0 > /sys/class/leds/led_blue/brightness
root@rzboard:~# echo heartbeat > /sys/class/leds/led_blue/trigger
```

LED output red:

```
root@rzboard:~# echo 0 > /sys/class/leds/led_blue/brightness
root@rzboard:~# echo 1 > /sys/class/leds/led_red/brightness
root@rzboard:~# echo 0 > /sys/class/leds/led_red/brightness
```

LED output green:

```
root@rzboard:~# echo 0 > /sys/class/leds/led_blue/brightness
root@rzboard:~# echo 1 > /sys/class/leds/led_green/brightness
root@rzboard:~# echo 0 > /sys/class/leds/led_green/brightness
```

3.3 Button Switches

There are two push-button switches on RzBoard, S1 is the power button and S2 is the user button. We can use following procedure to test these button switches.

Test PWR button S1

When system is on, press PWR button S1 for 3 seconds, the system will shut down.
Press the PWR button S1 again for 3 seconds and the system will reboot.

Test USER button S2

```
root@rzboard:~# evtest
No device specified, trying to scan all of /dev/input/event*
Available devices:
/dev/input/event0:          keys
Select the device event number [0-0]: 0
Input driver version is 1.0.1
Input device ID: bus 0x19 vendor 0x1 product 0x1 version 0x100
Input device name: "keys"
Supported events:
  Event type 0 (EV_SYN)
  Event type 1 (EV_KEY)
    Event code 2 (KEY_1)
Properties:
Testing ... (interrupt to exit)
Event: time 1600620566.1600620566, type 1 (EV_KEY), code 2 (KEY_1), value 1
Event: time 1600620566.1600620566, ---------- SYN_REPORT ------------
Event: time 1600620566.1600620566, type 1 (EV_KEY), code 2 (KEY_1), value 0
Event: time 1600620566.1600620566, ---------- SYN_REPORT ------------
Event: time 1600620570.1600620570, type 1 (EV_KEY), code 2 (KEY_1), value 1
Event: time 1600620570.1600620570, ---------- SYN_REPORT ------------
Event: time 1600620570.1600620570, type 1 (EV_KEY), code 2 (KEY_1), value 0
Event: time 1600620570.1600620570, ---------- SYN_REPORT ------------
Event: time 1600620606.1600620606, type 1 (EV_KEY), code 2 (KEY_1), value 1
```

```
Event: time 1600620606.1600620606, ---------- SYN_REPORT ------------
Event: time 1600620606.1600620606, type 1 (EV_KEY), code 2 (KEY_1), value 0
Event: time 1600620606.1600620606, ---------- SYN_REPORT ------------
Event: time 1600620609.1600620609, type 1 (EV_KEY), code 2 (KEY_1), value 1
Event: time 1600620609.1600620609, ---------- SYN_REPORT ------------
```

Use "**Ctrl+C**" to exit this test.

3.4 Display Output

RzBoard supports MIPI-DSI and HDMI screen.

Users can connect the screen to the board before boot up the system according to the following table. When the system boot up, the screen will print the related startup message and login UI. Users can connect keyboard to login the RzBoard file system.

Screen Type	Screen Resolution	Interface
MIPI-DSI	720*1280	J5 (MIPI-DSI)
MIPI to HDMI	Adjust to the screen size	J13 (microHDMI)

3.4.1 MIPI-DSI Screen

If you choose MIPI-DSI display and it's model# is *PH720128T003*, you should edit **uEnv.txt** as follows:

```
#enable_overlay_disp=mipi
fdt_extra_overlays=rzboard-mipi-ph720128t003.dtbo
```

If you choose MIPI-DSI display and it's model# is *PH720128T005*, you should edit uEnv.txt as follows:

```
enable_overlay_disp=mipi
#fdt_extra_overlays=1.dtbo 2.dtbo 3.dtbo
```

MIPI-DSI supports adjustment of the LCD backlight brightness. The backlight brightness has a range from 0 to 9, where 9 is highest brightness, 0 is the lowest.

Execute the following instructions on the serial terminal to implement the backlight test:

```
root@rzboard:~# echo 7 > /sys/class/backlight/backlight/brightness
```

3.4.2 MIPI To HDMI Screen

RzBoard also supports MIPI to HDMI screen, Choose MIPI to HDMI screen, the *enable_overlay_disp* value should be:

```
enable_overlay_disp=hdmi
```

3.5 Audio

RzBoard's audio subsystem, includes audio codec, stereo headphone jack I/O, HDMI audio I/O, as well as USB and Bluetooth based audio I/O.

3.5.1 Check Audio Device IDs

Before playing or recording an audio interface, you should check the device ID.

Us the **aplay -l** and **arecord -l** commands to list the audio playback- and record- device IDs.

By default, you should see the following devices:

```
root@rzboard:~# aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: audioda7212 [audio-da7212], device 0: ssi-dai-da7213-hifi da7213-hifi-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: litecodec [lite-codec], device 0: ssi-dai-avt-lite-codec-pcm-wb avt-lite-codec-pcm-wb-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 2: hdmisoundcard [hdmi-sound-card], device 0: ssi-dai-i2s-hifi i2s-hifi-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0

root@rzboard:~# arecord -l
**** List of CAPTURE Hardware Devices ****
card 0: audioda7212 [audio-da7212], device 0: ssi-dai-da7213-hifi da7213-hifi-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: litecodec [lite-codec], device 0: ssi-dai-avt-lite-codec-pcm-wb avt-lite-codec-pcm-wb-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

You can modify the default sound card by editing **/etc/asound.conf** :

```
root@rzboard:~# vi /etc/asound.conf
# use da7212 as default sound card
defaults.pcm.card 1
defaults.pcm.device 0
defaults.ctl.card 1
```

3.5.1.1 On-board Audio Codec

DA7212 is the on-board audio codec on RzBoard, It is also the default audio device of the RzBoard, will be enabled automatically when the RzBoard starts up. Use command **arecord -l** and **aplay -l** to check that the device id is 0.

3.5.1.2 Stereo Jack Analog Audio I/O

J16 is an extension audio output interface of RzBoard. To enable the extension audio output interface, use the *enable_overlay_audio* option in uEnv.txt like following:

```
enable_overlay_audio=1
```

Connect an audio device such as 3.5mm headset to J16 to use it.
Use the command **aplay -l** to check the device ID.

3.5.1.3 USB Audio Device

RzBoard can support a USB audio device (which do not need specific driver) to play audio. You can record and play audio from USB audio device. Use command **arecord -l** and **aplay -l** to check that the device id is 1.

```
root@rzboard:~# aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: audioda7212 [audio-da7212], device 0: ssi-dai-da7213-hifi da7213-hifi-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: Seri [Plantronics Blackwire 3215 Seri], device 0: USB Audio [USB Audio]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
root@rzboard:~# aplay -L
null
    Discard all samples (playback) or generate zero samples (capture)
default:CARD=audioda7212
    audio-da7212,
    Default Audio Device
sysdefault:CARD=audioda7212
    audio-da7212,
    Default Audio Device
default:CARD=Seri
    Plantronics Blackwire 3215 Seri, USB Audio
    Default Audio Device
sysdefault:CARD=Seri
    Plantronics Blackwire 3215 Seri, USB Audio
    Default Audio Device
```

```
front:CARD=Seri,DEV=0
    Plantronics Blackwire 3215 Seri, USB Audio
    Front speakers
surround21:CARD=Seri,DEV=0
    Plantronics Blackwire 3215 Seri, USB Audio
    2.1 Surround output to Front and Subwoofer speakers
surround40:CARD=Seri,DEV=0
    Plantronics Blackwire 3215 Seri, USB Audio
    4.0 Surround output to Front and Rear speakers
surround41:CARD=Seri,DEV=0
    Plantronics Blackwire 3215 Seri, USB Audio
    4.1 Surround output to Front, Rear and Subwoofer speakers
surround50:CARD=Seri,DEV=0
    Plantronics Blackwire 3215 Seri, USB Audio
    5.0 Surround output to Front, Center and Rear speakers
surround51:CARD=Seri,DEV=0
    Plantronics Blackwire 3215 Seri, USB Audio
    5.1 Surround output to Front, Center, Rear and Subwoofer speakers
surround71:CARD=Seri,DEV=0
    Plantronics Blackwire 3215 Seri, USB Audio
    7.1 Surround output to Front, Center, Side, Rear and Woofer speakers
iec958:CARD=Seri,DEV=0
    Plantronics Blackwire 3215 Seri, USB Audio
    IEC958 (S/PDIF) Digital Audio Output
```

3.5.2 Record Audio

Use the following command to record audio to an audio.wav file:

```
root@rzboard:~# arecord -f S16_LE -r 48000 -c 2 -Dhw:0 audio_test.wav
```

Note: Press **Ctrl+C** to exit recording.

In the above command:

S16_LE = audio format,
-r 48000 = sample rate of the audio file (48KHz),
-c 2 = 2 channel audio recording,
-Dhw:0 = use audio card 0 to record (device id of the codec-connected MIC),
Use command **arecord -l** and **aplay -l** to check the device ID.

Change those parameters according to your device.

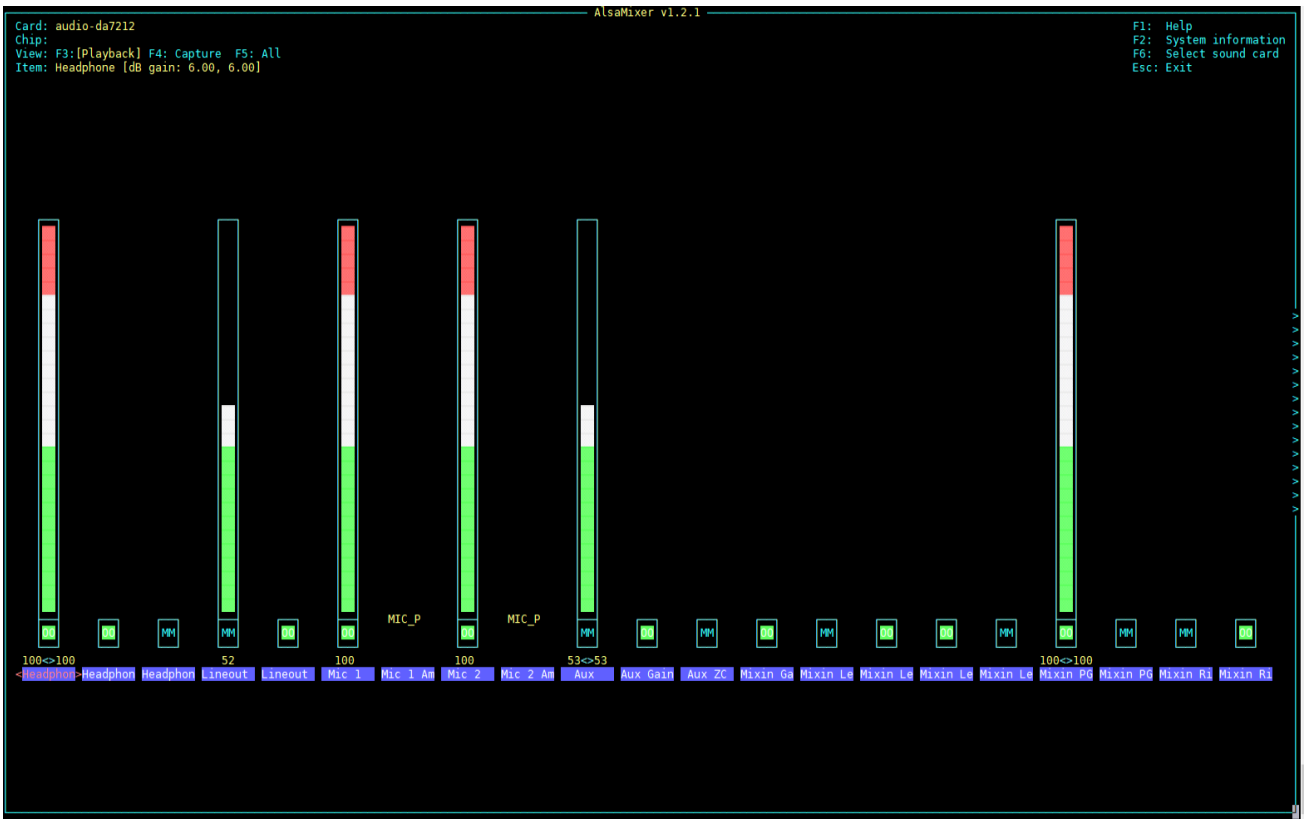
After recording, you can play the recorded audio file with the following command to verify.

```
root@rzboard:~# aplay audio_test.wav
```

To adjust the level of the audio recording, use the following command to open the ALSA mixer GUI

```
root@rzboard:~# alsamixer -c1
```

Use the Up-down-Left-Right button to adjust the volume of different channel, press Esc button to exit.



3.5.3 Play Audio File

```
root@rzboard:~# aplay audio_test.wav
Playing WAVE ' audio_test.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
^CAborted by signal Interrupt...
root@rzboard:~# gst-play-1.0 audio_test1.mp3
Press 'k' to see a list of keyboard shortcuts.
Now playing /home/root/ audio_test1.mp3
Redistribute latency...
0:00:17.6 / 0:03:28.5
```

aplay command supports .wav format audio files in,
gst-play-1.0 command supports wav, mp3 and aac formats

When using above command. Audio will play from the default device (on-board audio output interface)

To play the audio from a specific device, use the following:

```
root@rzboard:~# aplay -Dhw:1 audio_test.wav
Playing WAVE 'audio_test.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
^[^[^[^CAborted by signal Interrupt...
```

Here we use **-Dhw:1**, which means use audio card 1 to play the audio.

3.6 Video

This Yocto system supports playback of video files in mp4 format, with maximum resolution of 1080p
Select one of the following four commands and enter it in the serial terminal to play:

```
root@rzboard:~# gst-play-1.0 3b5e1066bf4ed2e142824231cf1a7017.mp4
Press 'k' to see a list of keyboard shortcuts.
Now playing /home/root/3b5e1066bf4ed2e142824231cf1a7017.mp4
[ 1614.627514] alloc_contig_range: [580b8, 580bf) PFNs busy
[ 1614.687575] alloc_contig_range: [5a200, 5a611) PFNs busy
[ 1614.696017] alloc_contig_range: [5a400, 5a811) PFNs busy

** (gst-play-1.0:483): CRITICAL **: 11:10:46.710:
file ../gst-plugins-base-1.16.3/gst-libs/gst/audio/gstaudioringbuffer.c: line 2048
(gst_audio_ring_buffer_set_channel_positions): should not be reached
Redistribute latency...
ts:1600600246.9734124  level:0x00010000  func:OmxrMcApiProxy_UseEGLImage(1212)
tid:500mes:This function is not implemented
eglDestroyImage not found
eglDestroyImage not found
eglDestroyImage not found
eglDestroyImage not found
NOTE: The GFX library has the time limitation by reason of an evaluation module.0:00:00.0 /
0:0[ 1617.433659] alloc_contig_range: [5a400, 5a5fe) PFNs busy
[ 1617.495800] alloc_contig_range: [5a400, 5a5fe) PFNs busy
WARNING A lot of buffers are being dropped.
WARNING debug information: ../gstreamer-1.16.3/libs/gst/base/gstbasesink.c(3005):
gst_base_sink_is_too_late ():
/GstPlayBin:playbin/GstPlaySink:playsink/GstBin:vbin/GstGLImageSinkBin:glimagesinkbin0/Gst
GLImageSink:sink:
There may be a timestamping problem, or this computer is too slow.

root@rzboard:~# gst-launch-1.0 playbin uri=file:///home/root/big_buck_bunny.mp4
Setting pipeline to PAUSED ...
Pipeline is PREROLLING ...
```

```
Got context from element 'sink': gst.gl.GLDisplay=context,
gst.gl.GLDisplay=(GstGLDisplay)"\(GstGLDisplayWayland\)\ gldisplaywayland0";
[ 1885.953189] alloc_contig_range: [580b8, 580bf) PFNs busy
Got context from element 'playsink': gst.gl.GLDisplay=context,
gst.gl.GLDisplay=(GstGLDisplay)"\(GstGLDisplayWayland\)\ gldisplaywayland0";
Redistribute latency...
ts:1600600518.272279      level:0x00010000          func:OmxrMcApiProxy_UseEGLImage(1212)
tid:537mes:This function is not implemented
eglDestroyImage not found
eglDestroyImage not found
eglDestroyImage not found
eglDestroyImage not found
NOTE: The GFX library has the time limitation by reason of an evaluation module.Pipeline is
PREROLLED ...
Setting pipeline to PLAYING ...
New clock: GstAudioSinkClock
[ 1888.732602] alloc_contig_range: [5a500, 5a5e1) PFNs busy
Got EOS from element "playbin0".
Execution ended after 0:01:00.127540445
Setting pipeline to NULL ...
eglDestroyImage not found
Freeing pipeline ...
```

3.7 Camera

RzBoard can support a USB camera or MIPI-CSI camera. This section describes how to preview, capture photos and record video from the command line.

To use the MIPI-CSI camera, the ***enable_overlay_camera*** value should be set:

```
enable_overlay_camera=ov5640
```

Note: To enable the camera preview on the desktop, it is recommended that the "*enable_overlay_mipi*" option should be set in **uEnv.txt**.

3.7.1 Enable the CSI-2 Module

According to the usage reference document (https://renesas.info/wiki/RZ-G/RZ-G2L_SMARC#Using_the_Coral_MIPI_Camera) of Renesas' MIPI camera, we know that prior to using the camera, the media-ctl command provided in the v4l-utils package must be used to configure the MIPI CSI-2 module, otherwise the OV5640 will not work.
Next we use the following command to enable the CSI-2 module.

```
root@rzboard:~# ls /dev/media*
```

```
/dev/media0
root@rzboard:~# media-ctl -d /dev/media0 -r
root@rzboard:~# media-ctl -d /dev/media0 -l "'rzg2l_csi2 10830400.csi2':1 -> 'CRU output':0 [1]"
```

3.7.2 Select OV5640 Camera and Set its Resolution

Use the following instruction to select OV5640 Camera and set its resolution.

```
root@rzboard:~# media-ctl -d /dev/media0 -V "'rzg2l_csi2 10830400.csi2':1 [fmt:UYVY8_2X8/1920x1080 field:none]"
root@rzboard:~# media-ctl -d /dev/media0 -V "'ov5640 0-003c':0 [fmt:UYVY8_2X8/1920x1080 field:none]"
```

3.7.3 Take Photo

Use the following instruction to take a photo and saved to specific location.

```
root@rzboard:~# yavta -c1 –F[filename] -s [resolution]    [video]
```

or

```
gst-launch-1.0 v4l2src device=[video] num-buffers=1 ! jpegenc ! filesink location=[filename]
```

In above command, replace [video] to the camera device ID, [filename] to the path and name of saved file, [resolution] to the resolution.

For example:

```
root@rzboard:~# yavta -c1 -Fyavta_video_1920x1080_1.yuv -s 1920x1080 /dev/video0
```

or

```
root@rzboard:~# gst-launch-1.0 v4l2src device=/dev/video0 num-buffers=1 !
'video/x-raw,format=UYVY,width=1920,height=1080' ! jpegenc ! filesink
location=ov5640_capture.jpg
```

Use the follow following command to view this photo directly:

```
root@rzboard:~# gst-launch-1.0 v4l2src device=/dev/video0 ! videoconvert ! waylandsink
```

Or copy the photo to other device, such as computer to display it.

3.7.4 Record Video

Use the following instruction to record a video and saved to specific location.

```
root@rzboard:~# gst-launch-1.0 -e v4l2src device=/dev/video0 num-buffers=300 !
video/x-raw,format=YUY2,framerate=30/1,width=640,height=480 ! videoconvert ! x264enc !
video/x-h264, profile=baseline ! mp4mux ! filesink location=output.mp4
```

In above command, modify the camera device ID, the width and height of the video, the path and name of saved file, etc. The video file can be copy to other device, such as computer to display, or use gst-play-1.0 to display it on the screen directly.

```
root@rzboard:~# gst-play-1.0 output.mp4
```

3.8 Gigabit Ethernet Interface

Connect the network cable to J7, enter the following instructions to set the IP address:

The below IP address are example, replace it with your real network environment

3.8.1 Network Test

After connecting the network cable, it will automatically obtain the IP by default. You can use the ifconfig command to view the IP information and use the following command to perform the network test:

```
root@rzboard:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 22:31:56:BB:B0:08
          inet addr:192.168.1.99  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:32706 errors:0 dropped:6081 overruns:0 frame:0
          TX packets:2829 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3753830 (3.5 MiB)  TX bytes:282218 (275.6 KiB)
          Interrupt:92 DMA chan:ff
```

3.8.2 Set Static IP

If you need to set a static IP, execute the following 2 steps:

1. Set the static MAC for the Board: modify the ethaddr value in uEnv.txt.

Use **nano** or **vi** command to modify the **uEnv.txt**.

```
root@rzboard:/run/media/mmcblk0p1# vi /boot/uEnv.txt
```

After the modification, execute *sync* and *reboot* command to make it effect.

Then check the configuration of **eth0**

```
root@rzboard:/run/media/mmcblk0p1# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr AA:BB:CC:DD:EE:FE
          inet addr:192.168.1.99  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:45527 errors:0 dropped:8527 overruns:0 frame:0
          TX packets:1257 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5108847 (4.8 MiB)  TX bytes:79467 (77.6 KiB)
          Interrupt:92 DMA chan:ff

root@rzboard:/run/media/mmcblk0p1# ping baidu.com
PING baidu.com (220.181.38.251): 56 data bytes
64 bytes from 220.181.38.251: seq=0 ttl=50 time=23.486 ms
64 bytes from 220.181.38.251: seq=1 ttl=50 time=22.838 ms
64 bytes from 220.181.38.251: seq=2 ttl=50 time=23.138 ms
64 bytes from 220.181.38.251: seq=3 ttl=50 time=23.287 ms
64 bytes from 220.181.38.251: seq=4 ttl=50 time=23.166 ms
```

2. Set Static IP info:

```
root@rzboard:~# vi /etc/systemd/network/01-eth0.network
[Match]
Name=eth0

[Network]
Address=192.168.1.99/24
Gateway=192.168.1.1
DNS=114.114.114.114
DNS=223.6.6.6

root@rzboard:~# systemctl restart systemd-networkd
```

In above command, replace the IP address, router, DNS with your real network environment. Execute **sync** after the modification, then **reboot** the system to make it effect.

3.8.3 Set Dynamic IP

```
root@rzboard:~# vi /etc/systemd/network/01-eth0.network
```

```
[Match]
Name=eth0

[Network]
DHCP=yes
root@rzboard:~# systemctl restart systemd-networkd
```

3.9 Storage

RzBoard supports on-board eMMC and SD Card interface, it can boot from SD Card ***or*** eMMC.

Note： Due to the SD card and eMMC flash share the same hardware interface. Therefore, the system starts from SD card if it detects an SD card, and starts from eMMC if it does not， cannot use SD card and eMMC at the same time.

Use **lsblk** command to list all available block devices in system:

```
root@rzboard:~# lsblk
```

3.9.1 SD Card

The storage node for SD Card is **/dev/mmcblk0**.

To boot from SD Card, Insert the SD card into the card slot before power on the board.

3.9.2 eMMC

The size of on-board eMMC is 32GB.

The storage node for eMMC is **/dev/mmcblk0**.

To boot from eMMC, *SD card must be removed before power-up of the board*.

3.10 USB 2.0 Interface

RzBoard supports two USB 2.0 Host interfaces.

3.10.1 USB Host

Insert a U-disk, serial terminal will display the disk information:

```
[  108.102562] usb 1-1.3: new high-speed USB device number 3 using ci_hdrc
```

```
[    108.154161] usb-storage 1-1.3:1.0: USB Mass Storage device detected
[    108.161226] scsi host0: usb-storage 1-1.3:1.0
[    109.184992] scsi 0:0:0:0: Direct-Access     Kingston DataTraveler 3.0     PQ: 0 ANSI: 6
[    109.196299] sd 0:0:0:0: [sda] 30218842 512-byte logical blocks: (15.5 GB/14.4 GiB)
[    109.204707] sd 0:0:0:0: [sda] Write Protect is off
[    109.210058] sd 0:0:0:0: [sda] Write cache: disabled, read cache: enabled, doesn't support DPO or FUA
[    109.249451]  sda: sda1
[    109.256908] sd 0:0:0:0: [sda] Attached SCSI removable disk
```

Execute the following instructions on the serial terminal:

```
root@rzboard:~# ls /dev/sd*
/dev/sda  /dev/sda1  /dev/sda2
```

The storage node for U disk is /dev/sda1, uses could mount the storage device to the file system to read and write data.

RzBoard also supports other USB device such as key board, mouse, Camera, etc.

3.10.2 USB OTG

There is a USB OTG connecter(J3) on RzBoard, you can connect a USB device by USB OTG cable.

When inserting a USB device, serial terminal will display the device information:

```
root@rzboard:~# [ 1050.341207] usb 3-1: USB disconnect, device number 2
[ 1054.790313] usb 1-1: new high-speed USB device number 4 using ehci-platform
[ 1054.952602] usb-storage 1-1:1.0: USB Mass Storage device detected
[ 1054.959105] scsi host0: usb-storage 1-1:1.0
[ 1056.602525] scsi 0:0:0:0: Direct-Access     SD Card Reader    1.00 PQ: 0 ANSI: 6
[ 1056.611640] sd 0:0:0:0: [sda] 15529984 512-byte logical blocks: (7.95 GB/7.41 GiB)
[ 1056.623642] sd 0:0:0:0: [sda] Write Protect is off
[ 1056.630560] sd 0:0:0:0: [sda] No Caching mode page found
[ 1056.636098] sd 0:0:0:0: [sda] Assuming drive cache: write through
[ 1056.648013]  sda: sda1 sda2
[ 1056.655146] sd 0:0:0:0: [sda] Attached SCSI removable disk
[ 1057.922206] EXT4-fs (sda2): mounted filesystem with ordered data mode. Opts: (null)
[ 1058.074230] FAT-fs (sda1): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.
```

Execute the following instructions on the serial terminal:

```
root@rzboard:~# lsusb
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

```
Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 001 Device 002: ID 067b:2731 Prolific Technology, Inc. USB SD Card Reader
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 003 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
root@rzboard:~# ls /dev/sd*
/dev/sda    /dev/sda1    /dev/sda2
```

The storage node for U disk is /dev/sda, uses could mount the storage device to the file system to read and write data.

RzBoard also supports other USB devices such as key board, mouse, Camera, etc.

Change U disk to USB mouse:

```
root@rzboard:~# [    869.569244] usb 1-1: USB disconnect, device number 2
[    873.814314] usb 3-1: new low-speed USB device number 2 using ohci-platform
[    874.064980] input: PixArt HP USB Optical Mouse as
/devices/platform/soc/11c50000.usb/usb3/3-1/3-1:1.0/0003:03F0:094A.0001/input/input1
[    874.077338] hid-generic 0003:03F0:094A.0001: input: USB HID v1.11 Mouse [PixArt HP USB
Optical Mouse] on usb-11c50000.usb-1/input0
[    874.131142] mousedev: PS/2 mouse device common for all mice

root@rzboard:~# ls /dev/sd*
ls: cannot access /dev/sd*: No such file or directory
root@rzboard:~# lsusb
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 003 Device 002: ID 03f0:094a HP, Inc Optical Mouse [672662-001]
Bus 003 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

3.11 Wi-Fi

The on-board Wi-Fi module supports 2.4G/5G network.

3.11.1 Enable Wi-Fi

User can run the following commands to start Wi-Fi:

```
root@rzboard:~# ifconfig mlan0 up
```

3.11.2 Connect Wi-Fi

Execute the following instructions on the serial terminal to search Wi-Fi network:

```
root@rzboard:~# iwlist mlan0 scan | grep SSID
                    ESSID:"MAX8DEV"
                    ESSID:"MAX8DEV_5G"
```

It prints the information for all available network.

Configure SSID and SSID_PASSWD with the following command: (take "MAX8DEV" as an example)

```
root@rzboard:~# wpa_passphrase "MAX8DEV" "12345678" >> /etc/wpa_supplicant.conf
```

Or edit ***/etc/wpa_supplicant.conf*** directly and append the following parameters:

```
network={
    ssid="MAX8DEV"
    psk="12345678"
}
```

Run the following command to start the Access Point:

```
root@rzboard:~# wpa_supplicant -B -i mlan0 -c /etc/wpa_supplicant.conf
```

Command output example:

```
root@rzboard:~# wpa_supplicant -B -i mlan0 -c /etc/wpa_supplicant.conf
Successfully initialized wpa_supplicant
rfkill: Cannot open RFKILL control device
[ 2324.243090] wlan: mlan0 START SCAN
[ 2328.723148] wlan: SCAN COMPLETED: scanned AP count=2
[ 2328.761226] wlan: Connected to bssid 80:XX:XX:XX:f6:d2 successfully
[ 2328.867423] mlan0:
[ 2328.867433] wlan: Send EAPOL pkt to 80:XX:XX:XX:f6:d2
[ 2328.879065] mlan0:
```

```
[ 2328.879074] wlan: Send EAPOL pkt to 80:XX:XX:XX:f6:d2
[ 2328.891195] woal_cfg80211_set_rekey_data return: gtk_rekey_offload is DISABLE
[ 2330.875079] wlan: mlan0 START SCAN
[ 2340.360108] wlan: SCAN COMPLETED: scanned AP count=2
```

Run the command to get the IP address:

```
root@rzboard:~# udhcpc -i mlan0 -n -R
udhcpc: started, v1.31.1
udhcpc: sending discover
udhcpc: sending discover
udhcpc: sending discover
udhcpc: sending select for 192.168.1.240
udhcpc: lease of 192.168.2.240 obtained, lease time 86400
/etc/udhcpc.d/50default: Adding DNS 114.114.114.114
/etc/udhcpc.d/50default: Adding DNS 8.8.8.8
[ 2484.528529] ravb 11c20000.ethernet eth0: Link is Down
root@rzboard:~# ifconfig mlan0
```

Test Wi-Fi network with **ping** command:

```
root@rzboard:~# ping www.baidu.com -I mlan0
PING www.baidu.com (110.242.68.4): 56 data bytes
64 bytes from 110.242.68.4: seq=0 ttl=54 time=26.614 ms
64 bytes from 110.242.68.4: seq=1 ttl=54 time=28.111 ms
64 bytes from 110.242.68.4: seq=2 ttl=54 time=27.055 ms
64 bytes from 110.242.68.4: seq=4 ttl=54 time=27.584 ms
64 bytes from 110.242.68.4: seq=5 ttl=54 time=25.901 ms
```

3.11.3 Wi-Fi Hotspot

Use the following steps to configure and start the 2.4 GHz Access Point from the wireless module.

Make sure the Wi-Fi is disconnected:

```
root@rzboard:~# killall wpa_supplicant
root@rzboard:~# killall hostapd
```

then use the following steps to set up Wi-Fi hotspot.

Edit the configuration file for hostapd:

```
root@rzboard:~# vi /etc/hostapd-2.4g.conf
```

Parameter values in the configuration file:

```
interface=uap0
# specify the band: hw_mode=g (2.4 GHz) and hw_mode=a (5 GHz)
hw_mode=g
channel=1
country_code=US
ssid=MY_HOSTAP
ieee80211n=1
```

If you want to configure WPA2 for the AP using open source supplicant, need to add the following additional lines:

```
wpa=2
wpa_key_mgmt=WPA-PSK
rsn_pairwise=CCMP
wpa_passphrase=123456789
```

Note: You can modify your **ssid** and **wpa_passphrase** in hostapd-2.4g.conf file.

Create the configuration file for udhcp server:

```
root@rzboard:~# vi /etc/udhcpd.conf
```

Add the following content to udhcpd.conf file:

```
interface uap0
start 192.168.5.10
end 192.168.5.100
opt router 192.168.5.1
opt dns 114.114.114.114 8.8.8.8
```

Note: The IP address 192.168.5.x can be modified at will but it must be consistent with its related IP.

Command to start the 2.4g GHz Access Point and start udhcp server to assign the IP address:

```
root@rzboard:~# ifconfig uap0 192.168.5.1 netmask 255.255.255.0 up
root@rzboard:~# udhcpd /etc/udhcpd.conf
root@rzboard:~# hostapd -B /etc/hostapd-2.4g.conf
```

At this time, you can use other devices to scan the access point **"*MY_HOSTAP*"**, and enter the password **"*123456789*"** to connect. After obtaining the IP address, the device will display a status of "Connected, no Internet".

If the Ethernet interface is connected to the Internet, you can use the following command to add packet forwarding rules so that the devices connected to the hotspot can access the Internet.

```
root@rzboard:~# echo 1 > /proc/sys/net/ipv4/ip_forward
root@rzboard:~# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
root@rzboard:~# iptables -A FORWARD -i eth0 -o uap0 -m state \
--state RELATED,ESTABLISHED -j ACCEPT
root@rzboard:~# iptables -A FORWARD -i uap0 -o eth0 -j ACCEPT
```

3.12 Bluetooth 5.0

The firmware binary file supports both Wi-Fi and Bluetooth over an SDIO interface, so user should enable Wi-Fi first (refer to Chapter 3.11.1).

Before using Bluetooth, we need to use the *hciattach* command to establish a data connection channel between the serial port and the Bluetooth protocol layer. This command is mainly used to initialize the Bluetooth device.:

```
root@rzboard:~# hciattach /dev/ttySC1 any 115200
Device setup complete
```

Use **hciconfig** to check the Bluetooth address:

```
root@rzboard:~# hciconfig hci0 up
root@rzboard:~# hciconfig hci0 version
hci0:   Type: Primary   Bus: UART
        BD Address: D4:53:83:C1:BD:13   ACL MTU: 1016:5   SCO MTU: 60:12
        HCI Version: 5.2 (0xb)   Revision: 0x8300
        LMP Version: 5.2 (0xb)   Subversion: 0x10d2
        Manufacturer: Marvell Technology Group Ltd. (72)
```

3.12.1 Connect Bluetooth Device

Use **bluetoothctl** to connect Bluetooth Device:

```
root@rzboard:~# bluetoothctl
[bluetooth]# power on
[bluetooth]# pairable on
[bluetooth]# agent on
[bluetooth]# default-agent
```

Make the RzBoard discoverable by other Bluetooth device:

```
[bluetooth]# discoverable on
```

Enable and Disable Scan:

```
[bluetooth]# scan on
[bluetooth]# scan off
```

Pair and connect the device:

```
[bluetooth]# pair E8:EC:A3:21:57:6C
[bluetooth]# trust E8:EC:A3:21:57:6C
[bluetooth]# connect E8:EC:A3:21:57:6C
```

Exit **bluetoothctl**.

```
[Mi Sports BT Earphones Basic]# quit
```

In above instructions, **E8:EC:A3:21:57:6C** is the address of the Bluetooth device, change it according to your device.

3.12.2 Send Files

Run the obexctl daemon and connect to the target Bluetooth device

```
root@rzboard:~# export $(dbus-launch)
root@rzboard:~# /usr/libexec/bluetooth/obexd -r /home/root -a -d & obexctl
[2] 568
[NEW] Client /org/bluez/obex
[obex]# connect 88:F5:6E:08:EC:26
[88:F5:6E:08:EC:26]# send /boot/uEnv.txt
Attempting to send /boot/uEnv.txt to /org/bluez/obex/client/session0
[NEW] Transfer /org/bluez/obex/client/session0/transfer1
Transfer /org/bluez/obex/client/session0/transfer1
        Status: queued
        Name: uEnv.txt
        Size: 183
        Filename: /boot/uEnv.txt
        Session: /org/bluez/obex/client/session0
```

3.13 UARTS

RzBoard supports two UART interfaces.

RzBoard (CPU)	Interface Type
UART0	UART TTL (Debug Interface)
UART2	UART TTL (on expansion connector)

3.13.1 UART 2

In the Yocto system, the node for UART2 is /dev/ttySC2. Users could also write their own applications to control the uart.

Use *enable_overlay_uart2* in **uEnv.txt** to allow UART2 to be selected:

```
enable_overlay_uart2=1
```

3.14 Pi HAT 40 Pin Expansion Interface

This chapter will provide the control methods of 40 Pin interface, include GPIO, I2C and SPI.

To use these peripheral interfaces on the 40-pins interface, enable the following options in **uEnv.txt** :

```
enable_overlay_gpio=1
enable_overlay_i2c=1
enable_overlay_spi=1
```

3.14.1 GPIO

System use /sys/class/gpio to control the GPIO pin, refer to the following table:

Table: GPIO# to Connector PIN# relationship

GPIO Number	PINMUX	Function	PIN #	PIN #	Function	PINMUX	GPIO Number
		3.3V	**1**	**2**	5V		
	I2C2	SDA1	**3**	**4**	5V		
	I2C2	SCL1	**5**	**6**	GND		
216	GPIO12_IO0	GPIO	**7**	**8**	UART_TX	UART2	
		GND	**9**	**10**	UART_RX	UART2	
507	GPIO48_IO3	GPIO	**11**	**12**	GPIO	GPIO17_IO1	**257**
506	GPIO48_IO2	GPIO	**13**	**14**	GND		
256	GPIO17_IO0	GPIO	**15**	**16**	GPIO	GPIO13_IO2	**226**
		3.3V	**17**	**18**	GPIO	GPIO14_IO0	**232**
	SPI1	MOSI	**19**	**20**	GND		
	SPI1	MISO	**21**	**22**	GPIO	GPIO39_IO1	**433**
	SPI1	SCLK	**23**	**24**	CE0	SPI1	
		GND	**25**	**26**	GPIO	GPIO0_IO1	**121**
233	GPIO14_IO1	GPIO	**27**	**28**	GPIO	GPIO46_IO3	**491**
459	GPIO42_IO3	GPIO	**29**	**30**	GND		
460	GPIO42_IO4	GPIO	**31**	**32**	GPIO	GPIO15_IO1	**241**
200	GPIO10_IO0	GPIO	**33**	**34**	GND		
193	GPIO9_IO1	GPIO	**35**	**36**	GPIO	GPIO48_IO4	**508**
225	GPIO13_IO1	GPIO	**37**	**38**	GPIO/ CAN0_RX	CAN0_RX	
		GND	**39**	**40**	GPIO/ CAN0_TX	CAN0_TX	

pinum = $group * $groupin + $pin + $pinbase *（where pinbase=120, groupin=8）*
Here we use **PIN35** as an example:

In above table, the GPIO Number of connector **PIN35** is calculated to be **193**
GPIO9_IO1 means group=**9**, pin=**1** for calculation of: **(9 x 8) + 1 + 120 = 193**

1. Set the function of Pin35 to be GPIO output.

```
root@rzboard:/sys/class# echo 193 >/sys/class/gpio/export
root@rzboard:/sys/class/gpio# echo out >/sys/class/gpio/gpio193/direction
```

2. Set the level of Pin35, 0 means low, 1 means high.

```
root@rzboard:/sys/class/gpio# echo 1 >/sys/class/gpio/gpio193/value
```

Measure the voltage of pin35, the result is 3.3V.

```
root@rzboard:/sys/class/gpio# echo 0 >/sys/class/gpio/gpio193/value
```

Measure the voltage of pin35, the result is 0V.

3.14.2 SPI

Add *enable_overlay_spi=1* to uEnv.txt, then execute **sync** and **reboot** command to make it effect.
Connect SPI_MOSI(#19) and SPI_MISO(#21),then execute *spidev_test*, the result:

```
root@rzboard:~# ./spidev_test -D /dev/spidev1.0 -v
spi mode: 0x0
bits per word: 8
max speed: 500000 Hz (500 kHz)
TX | FF FF FF FF FF FF 40 00 00 00 00 95 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
F0 0D    |......@.....................|
RX | FF FF FF FF FF FF 40 00 00 00 00 95 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
F0 0D    |......@.....................|
```

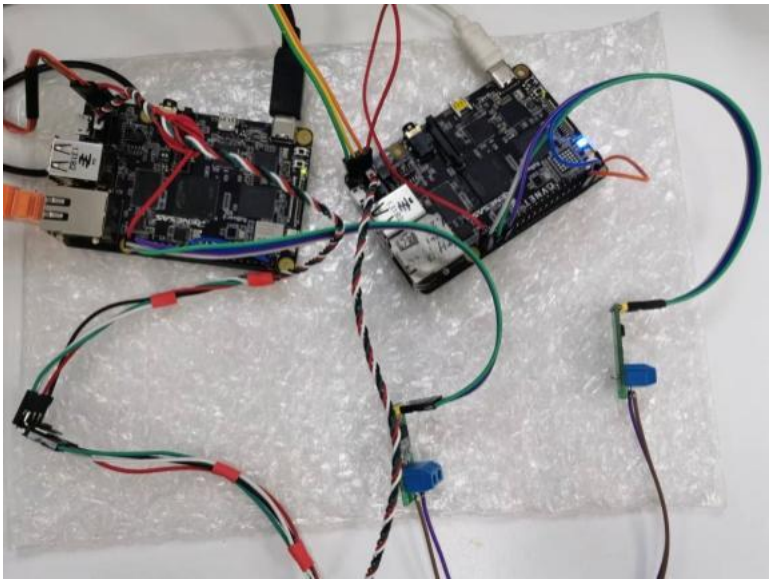
Disconnect SPI_MOSI(#19) and SPI_MISO(#21), then execute *spidev_test*, the result:

```
root@rzboard:~# ./spidev_test -D /dev/spidev1.0 -v
spi mode: 0x0
bits per word: 8
max speed: 500000 Hz (500 kHz)
TX | FF FF FF FF FF FF 40 00 00 00 00 95 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
F0 0D    |......@.....................|
RX | FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF    |............................|
```

3.14.3 CAN

When using CAN bus, the following should be noted regarding the onboard CAN interfaces:
The CAN interface on **J18** has a transceiver, and can be directly connected with other CAN interfaces;
The CAN interface on **J1** has **no** transceiver, it requires an external transceiver.

Shown below is an example test setup using both CAN interfaces on two RZBoards:



CANbus interfaces on RzBoard can work in 2.0 mode

Use *enable_overlay_can* in **uEnv.txt** to enable the CAN interface after RzBoard startup.

CAN 2.0 Test Commands and Results:

RzBoardA:

```
root@rzboard:~# ip link set can0 down
root@rzboard:~# ip link set can0 type can bitrate 500000
[ 1382.533140] rcar_canfd 10050000.can can0: bitrate error 0.2%
root@rzboard:~# ip link set can0 up
root@rzboard:~# candump can0
  can0  123  [7]  01 02 03 04 05 06 07
```

RzBoardB:

```
root@rzboard:~# ip link set can0 down
root@rzboard:~# ip link set can0 type can bitrate 500000
[ 1382.533140] rcar_canfd 10050000.can can0: bitrate error 0.2%
root@rzboard:~# ip link set can0 up
root@rzboard:~# cansend can0 123#01020304050607
```

Note: Testing of CAN interfaces on RzBoard confirmed reliable operation for bitrates up to 3.5 Mbps.

3.15 DRP-AI

RZ/V2L is equipped with a Cortex-A55 CPU and built-in "DRP-AI" AI accelerator core, for easy implementation of real-time AI inference and image processing functions on RzBoard.

Go to the **RZV2L_AI_Eva_**SW directory and execute the following commands to test the DRP-AI:

```
root@rzboard:~# cd RZV2L_AI_Eva_SW/
root@rzboard:~/RZV2L_AI_Eva_SW# ./start_app.sh I
IMAGE MODE
[INFO] Image Directory: bmp_img
[INFO] DRP-AI Object Files: resnet50_bmp
[START] Loading DRP-AI Data...
   [START] Loading resnet50_bmp/drp_desc.bin : size       0x1a0 at address 0x856d3f00
   [END]     Loading resnet50_bmp/drp_desc.bin
   [START] Loading resnet50_bmp/resnet50_bmp_drpcfg.mem : size 0x15d060 at address
0x855333c0
   [END]     Loading resnet50_bmp/resnet50_bmp_drpcfg.mem
   [START] Loading resnet50_bmp/drp_param.bin : size       0x120 at address 0x85690440
   [END]     Loading resnet50_bmp/drp_param.bin
   [START] Loading resnet50_bmp/aimac_desc.bin : size    0x43970 at address 0x85690580
   [END]     Loading resnet50_bmp/aimac_desc.bin
   [START] Loading resnet50_bmp/resnet50_bmp_weight.dat : size 0x30b5be0 at address
0x8247d7c0
   [END]     Loading resnet50_bmp/resnet50_bmp_weight.dat
[END] Loading DRP-AI Data : Total loading time    2.85 s
[bmp_img/sample.bmp]
1 images are loaded from bmp_img

Inference 1 -------------------------------------------
Input: bmp_img/sample.bmp
   DRP-AI processing time :    64.35 msec
   Output Binary                : resnet50_bmp_output/bmp_img/sample.bmp.bin
[INFO] 1 out of 1 images are processed.
[INFO] Output Log: resnet50_bmp_output/bmp_img/0920111653.log
```

With a USB camera connected to the board, you can test object recognition using DRP-AI
Run the following commands:

```
root@rzboard:~# cd app_demos/
root@rzboard:~/app_demos# ./demo.sh
*****************************************************************************************************************
*   Avnet                RZBoard V2L - DRP-AI demos (using camera video)                                          *
*                                                                                                                *
* a) Detection + Pose Estimation, skeletal 17-point overlay of person in box (HRNet)                             *
* b) Detection + Pose Estimation, skeletal overlays of 1-7 people (HRNet,TinyYOLOv2)                             *
* c) Object Classification, does not use bounding-box (ResNet50)                                                 *
* d) Object Classification, displays labeled boxes (Tiny YOLOv2)                                                 *
*                                                                                                                *
*****************************************************************************************************************
    Enter letter of AI demo to run...
>>
```

Then enter the applicable letter (a b c d) to select the DRP-AI demo that you want to run.

The processed camera image (with meta data and overlays) will be visible on the HDMI screen

3.16 Cortex-M33

On RzBoard, the User can enable the M33 core by editing uEnv.txt as follows:

```
root@rzboard:~# vi /boot/uEnv.txt

enable_overlay_cm33=1
#enable_overlay_uart2=1
```

After M33 is enabled, when u-boot bootsup, it will use fatload to load and run the Cortex-M33 firmware program. uart2 will be used as the Cortex-M33 core's debug serial port. We can test the **rpmsg** communication between Cortex-A55 and Cortex-M33, and output the test results from uart2.

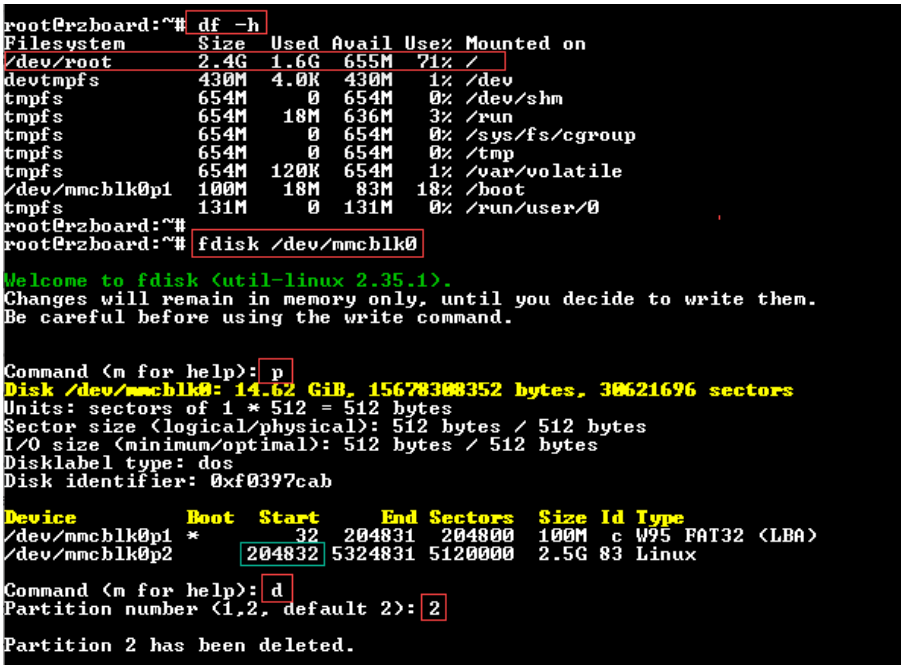
```
root@rzboard:~# rpmsg_sample_client 0
Successfully probed IPI device
metal: info:        metal_uio_dev_open: No IRQ for device 42f00000.rsctbl.
Successfully open uio device: 42f00000.rsctbl.
Successfully added memory device 42f00000.rsctbl.
metal: info:        metal_uio_dev_open: No IRQ for device 43000000.vring-ctl0.
Successfully open uio device: 43000000.vring-ctl0.
Successfully added memory device 43000000.vring-ctl0.
metal: info:        metal_uio_dev_open: No IRQ for device 43200000.vring-shm0.
Successfully open uio device: 43200000.vring-shm0.
Successfully added memory device 43200000.vring-shm0.
```

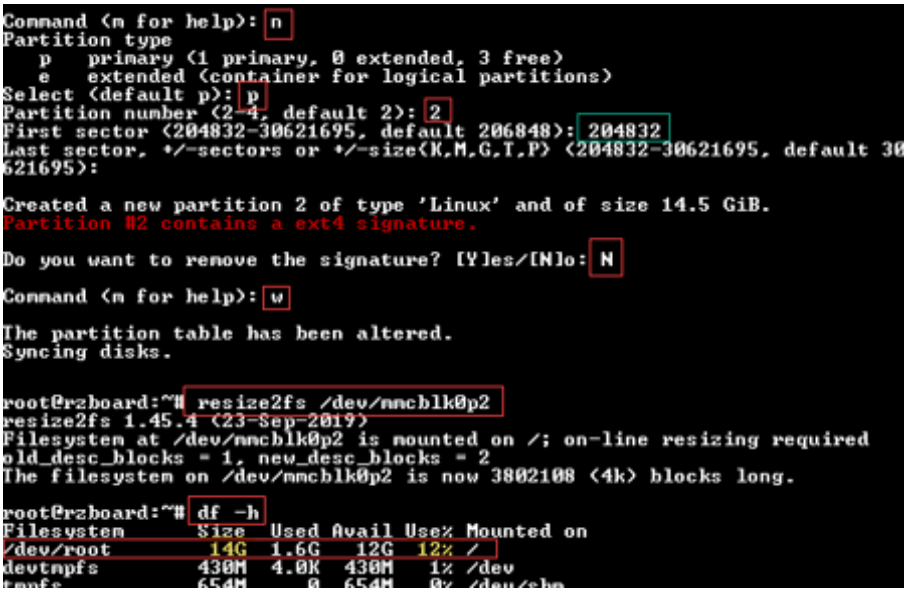
```
metal: info:      metal_uio_dev_open: No IRQ for device 42f01000.mhu-shm.
Successfully open uio device: 42f01000.mhu-shm.
Successfully added memory device 42f01000.mhu-shm.
Initialize remoteproc successfully.
creating remoteproc virtio
initializing rpmsg shared buffer pool
initializing rpmsg vdev
 1 - Send data to remote core, retrieve the echo and validate its integrity ..
Remote proc init.
RPMSG endpoint has created.
RPMSG service has created.
sending payload number 0 of size 17
echo test: sent : 17
 received payload number 0 of size 17
...
…
…
sending payload number 470 of size 487
echo test: sent : 487
 received payload number 470 of size 487
sending payload number 471 of size 488
echo test: sent : 488
 received payload number 471 of size 488

**********************************

 Test Results: Error count = 0

**********************************
Quitting application .. Echo test end
Stopping application...
```

3.17 Procedure to Increase eMMC Partition Size

As configured during manufacture, only a section of the 32GB eMMC is accessible.
Use the following steps to expand the rootfs partition in eMMC flash memory:

- Open a serial port connection to RZBoard's debug connector
- Boot Linux and login to the board with user: **root** and password: **avnet**
- Execute the command **fdisk /dev/mmcblk0**
- Make note of the **mmcblk0p2 start address** displayed on the screen
- Execute the following sequence of commands:
  p -> **d** -> **2** -> **n** -> **p** -> **2** -> **<mmcblk0p2 start address>** -> **enter** (to accept default) -> **N** -> **w**
- Now resize the partition using the entered settings: **resize2fs /dev/mmcblk0p2**





Chapter 4 Appendix

4.1 Hardware Documents

For hardware details please refer to:

- RzBoard Hardware User Guide:
- RzBoard Block Diagram

4.2 Software Documents

RzBoard supports Yocto Linux, for additional information, please refer to the following documents accessible from the RzBoard product page at https://www.avnet.me/rzboard

- RzBoard Linux Yocto Release Note

- RzBoard Linux Yocto User Manual
  - This document (describes how to reflash RZBoard and aspects of the BSP functionality)

- RzBoard Linux Yocto Development Guide
  - Detailed guidance on how to rebuild the Linux system image

4.3 Linux System Image and Application Development

4.3.1 Out of box System Image

At latest update of this document, the .manifest file for the 20221021 system image, lists the inclusion of a relatively wide range of software enablement, with Python, Gstreamer, DRP-AI examples, various editors, etc, all included within the file-system. Check the .manifest file in the image download for more details.

4.4 Contact Information

Product Page: https://www.avnet.me/rzboard