Contents

[1. Set testing environment 2](#_Toc35597516)

[2. Test script descript 3](#_Toc35597517)

[2.1 button check 3](#_Toc35597518)

[2.2 I2C inventory check 3](#_Toc35597519)

[2.3 PA fault check 4](#_Toc35597520)

[2.4 BT paring 5](#_Toc35597521)

[2.5 Set LED 5](#_Toc35597522)

[2.6 Get NTC 5](#_Toc35597523)

[2.7 GPIO set 6](#_Toc35597524)

[2.8 Audio set 6](#_Toc35597525)

[2.9 volume set 7](#_Toc35597526)

[2.10 Mic loopback 9](#_Toc35597527)

[2.11 burn in mode 9](#_Toc35597528)

[2.12 battery info 10](#_Toc35597529)

[2.13 NVRAM operate 10](#_Toc35597530)

# 1. Set testing environment

Step 1:

Use adb push factory test script to device

Exp:

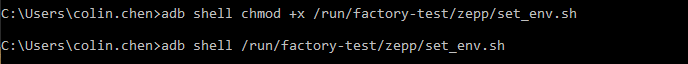


Run set\_env.sh

adb shell chmod +x <set\_env.sh path>

<set\_env.sh path>

Exp:



Then all the test script at project fold (exp: zepp) available.

# 2. Test script descript

Test script fold tree as below:

Factory test ------- main fold

| ---- platform ------- main script for whole platform

|----- zepp ------- project fold test script for specify project

|----- conf ------- configure file for specify script

For testing running the script at project fold (exp: zepp)

To configure specify test script: modify the .conf file at conf fold.

## 2.1 button check

Run button check script “button\_f\_check.sh” then push the button will get the test result. This script will wait 10S for pushing button.

All key define at conf file “key\_list.conf”. 1st row as key code using Hex form and 4bytes. 2nd row as key name

Exp:

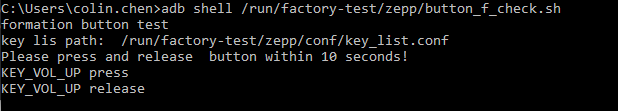
0066 KEY\_FORMATION

00a4 KEY\_PLAY\_PASUE

Test command:

adb shell /run/factory-test/zepp/button\_f\_check.sh

Exp:



## 2.2 I2C inventory check

Run “inventory.sh”

All I2C chip define at conf file “inventory.conf”. 1st row was i2c bus name , 2nd  row was I2C address, 3rd row was chip names.

**Notice: I2C address using 7bits address and Hex form. (exp: Address “0x40” will be “0x40/2” only use the digital part “20”)**

**Conf file exp:**

i2c-2 20 PA1-MA12070

i2c-2 21 PA2-MA12070

i2c-2 22 PA3-MA12070

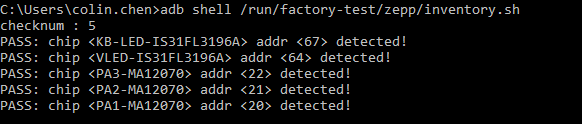
i2c-3 64 VLED-IS31FL3196A

i2c-3 67 KB-LED-IS31FL3196A

Test command :

adb shell /run/factory-test/zepp/inventory.sh

exp:



## 2.3 PA fault check

Run “get\_pa\_fault.sh”

PA fault pin define at file “pa\_fault.conf”. 1st row as gpio number. 2nd row as gpio name.

Exp:

81 PA1-fault

22 PA2-fault

23 PA3-fault

Test command :

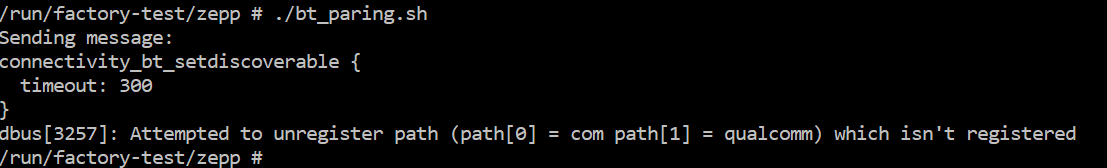
adb shell /run/factory-test/zepp/

Exp: (currently those gpio didn’t mount to system so this exp was fail exp)

## 2.4 BT paring

Run “bt\_paring.sh”

And BT will enter paring state. Using smart phone can search devices name.



## 2.5 Set LED

Run “set\_led.sh”

Led name and number define at file “led.conf”. 1st row as led name. 2nd row as led number

Exp:

led-formation1 1

led-formation2 2

led-voice-service1 3

led-voice-service2 4

all all

Parameter “all” support to operate all the available LED.

Command form:

set\_led.sh <Led name> <R brightness> <G brightness> <B brightness>

Brightness range: 0~255

Exp:



## 2.6 Get NTC

Run “get\_ntc.sh”

Thermal zone define at “ntc.conf”. 1st  row as thermal zone name. 2nd row as thermal zone number

Exp :

thermal\_zone4 4

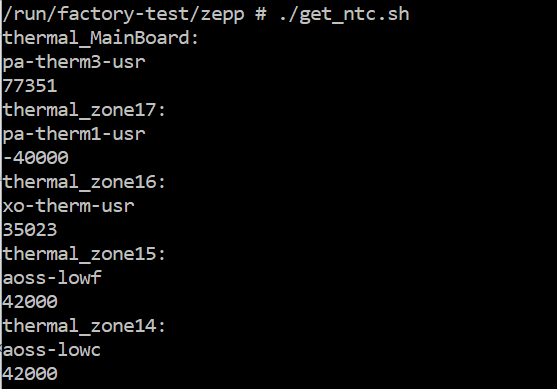
output form:

<thermal zone name>

<type> ----- name define at thermal driver

<thermal value> ------ value of micro Celsius degree

Exp:



## 2.7 GPIO set

Run “set\_gpio.sh”

GPIO list define at “gpio.conf” 1st row as gpio name. 2nd row as gpio number

Exp:

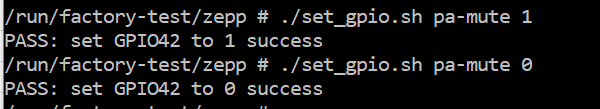
pa-mute 42

pa-en 40

Command form:

./set\_gpio.sh <gpio name> <value>

Exp:

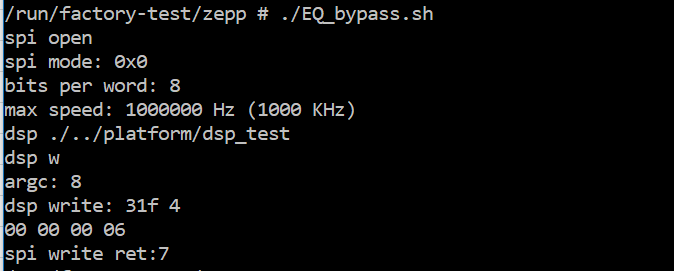


## 2.8 Audio set

Audio set using single script for each setting. Table as below:

|  |  |
| --- | --- |
| **script** | **function** |
| EQ\_bypass.sh | Set EQ (DSP) bypass |
| EQ\_enable.sh | Enable EQ |
| WF\_mute.sh | WF mute |
| WF\_enable.sh | WF enable |
| LMF\_mute.sh | LMF mute |
| LMF\_enable.sh | LMF enable |
| RMF\_mute.sh | RMF mute |
| RMF\_enable.sh | RMF enable |
| LHF\_mute.sh | LHF mute |
| LHF\_enable.sh | LHF enable |
| RHF\_mute.sh | RHF mute |
| RHF\_enable.sh | RHF enable |
| ~~AEC\_L\_mute.sh~~ | ~~Mute AEC L~~ |
| ~~AEC\_L\_enable.sh~~ | ~~Enable AEC L~~ |
| ~~AEC\_R\_mute.sh~~ | ~~Mute AEC R~~ |
| ~~AEC\_R\_enable.sh~~ | ~~Enable AEC R~~ |
| master\_mute.sh | Mute all |
| master\_enable.sh | Unmute all |
| set\_master\_source.sh | Set master source: soc/local/spdif |
| get\_dsp\_version.sh | Get DSP version |
| l\_tone1\_enable.sh | L channel tone1 enable |
| l\_tone1\_disable.sh | L channel tone1 disable |
| r\_tone1\_enable.sh | R channel tone1 enable |
| r\_tone1\_disable.sh | R channel tone1 disable |

Exp:



## 2.9 volume set

Run “set\_volume.sh”

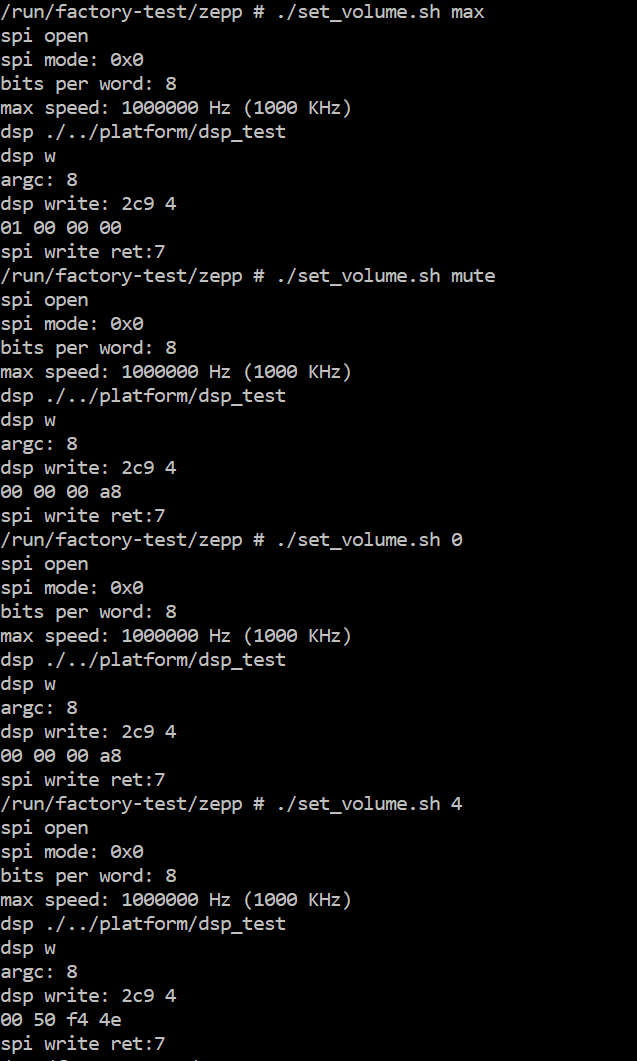
Volume detail data define at volume.conf from 1st line to last line was mute volume to max volume

Command form:

./set\_volumen.sh <value>

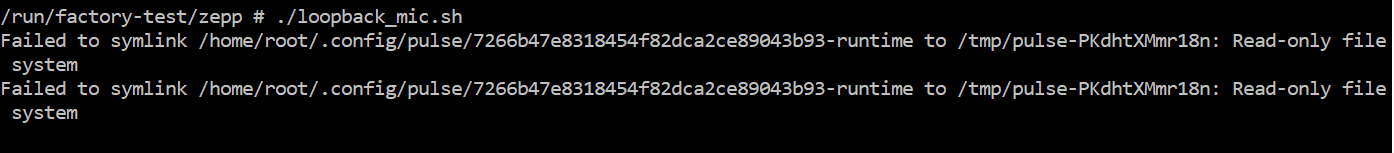
Value support number from 0~max volume level also “mute”, “max”

Exp:

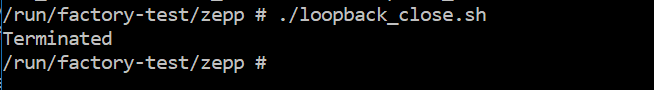


## 2.10 Mic loopback

Run “loopback\_mic.sh” to start mic loopback.



Run “loopback\_close.sh” to close mic loopback. Or force close the script to stop it.



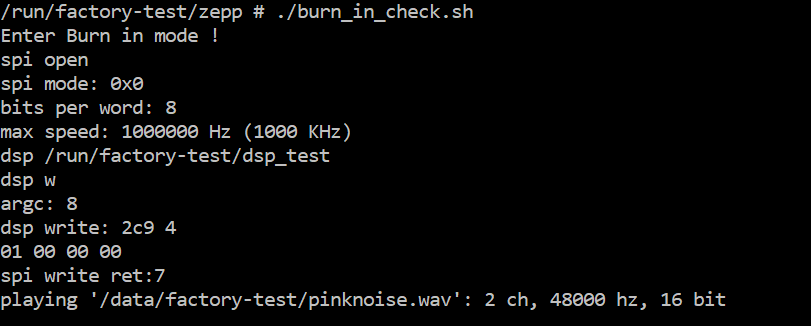
## 2.11 burn in mode

Run “set\_burn\_in.sh” to mark burn in mode.



This script will cope files that burn in mode need to /data

Run “burn\_in\_check.sh” will check the burn in mark, if it exist run the loop play pinknoise.



Currently auto running burn\_in\_check.sh can’t keep looping play pinknoise. When run on command line. It work.

## 2.12 battery info

Get battery capacity run “get\_batt\_capacity.sh”

Get battery temperature run “get\_batt\_temp.sh”

Get battery current run “get\_batt\_current.sh” . if the value was minus means charge out.

Get battery voltage run “get\_batt\_voltage.sh”

## 2.13 NVRAM operate

|  |  |
| --- | --- |
| **Script** | **Function** |
| nvram\_fs.sh | Formal NVRAM, after formal need to reboot to active it. |
| get\_assembly.sh | Get ASSEMBLY No. |
| Set\_assembly.sh | Set ASSEMBLY No. |
| get\_board\_version.sh | Get board version |
| get\_country\_code.sh | Get country code |
| set\_country\_code.sh | Set country code exp: US |
| get\_eth0\_mac\_addr.sh | Get ETH0 MAC address |
| set\_eth0\_mac\_addr.sh | Set ETH0 MAC address |
| get\_wlan0\_mac\_addr.sh | Get WLAN0 MAC address |
| set\_wlan0\_mac\_addr.sh | Set WLAN0 MAC address |
| get\_wlan1\_mac\_addr.sh | Get WLAN1 MAC address |
| set\_wlan1\_mac\_addr.sh | Set WLAN1 MAC address |
| get\_wlan2\_mac\_addr.sh | Get WLAN2 MAC address |
| set\_wlan2\_mac\_addr.sh | Set WLAN2 MAC address |
| get\_bt0\_mac\_addr.sh | Get BT0 MAC address |
| set\_bt0\_mac\_addr.sh | Set BT0 MAC address |
| get\_base\_mac\_addr.sh | Get BASE MAC address |
| set\_base\_mac\_addr.sh | Set BASE MAC address |
| get\_fp\_part.sh | Get FP part |
| Set\_fp\_part.sh | Set FP part |
| get\_unused0.sh | Get UNUSED0 |
| set\_unused0.sh | Set UNUSED0 |
| get\_unused1.sh | Get UNUSED1 |
| set\_unused1.sh | Set UNUSED1 |
| get\_model.sh | Get MODEL name |
| set\_model.sh | Set MODEL name |

Currently only wlan1 and bt0 mac address will set to modules. Others just store at NVRAM, and can be read. FW must have “nvram” tool.

Formal:

./set\_xxxx.sh <context>

Exp:

