Contents

[1. Set testing environment 2](#_Toc38647941)

[2. Test script descript 2](#_Toc38647942)

[2.1 button check 3](#_Toc38647943)

[2.2 I2C inventory check 3](#_Toc38647944)

[2.3 PA fault check 4](#_Toc38647945)

[2.4 BT paring 4](#_Toc38647946)

[2.5 Set LED 4](#_Toc38647947)

[2.6 Get NTC 5](#_Toc38647948)

[2.7 GPIO set 6](#_Toc38647949)

[2.8 Audio set 6](#_Toc38647950)

[2.9 volume set 7](#_Toc38647951)

[2.10 Mic loopback 9](#_Toc38647952)

[2.11 burn in mode 9](#_Toc38647953)

[2.12 battery info 10](#_Toc38647954)

[2.13 NVRAM operate 10](#_Toc38647955)

[2.14 Set Wlan0 MAC address 11](#_Toc38647956)

[2.15 Set BT MAC address 11](#_Toc38647957)

[2.15 Mic record 11](#_Toc38647958)

[2.16 Mic record close 11](#_Toc38647959)

[2.17 Audio play 12](#_Toc38647960)

[2.18 Audio play close 12](#_Toc38647961)

[2.19 Check audio ready 12](#_Toc38647962)

[2.20 Enter Wifi/BT FTM mode 12](#_Toc38647963)

[2.21 exit Wifi/BT FTM mode 12](#_Toc38647964)

[2.22 SPDIF On 13](#_Toc38647965)

[2.23 SPDIF OFF 13](#_Toc38647966)

# 1. Set testing environment

All script already install to “/etc/factory-test” both of it available for test.

Run “enter\_factory\_mode.sh” to set DUT environment for factory test. (Currently just disable LED manage)

When finished factory test run “exit\_factory\_mode.sh” to revert DUT environment for user mode.

**NOTICE: the command path have been change, but the exp. picture of this document will using the old one.**

# 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:

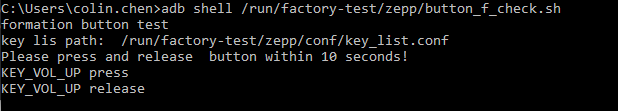
0066 KEY\_FORMATION

00a4 KEY\_PLAY\_PASUE

Test command:

adb shell /etc/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

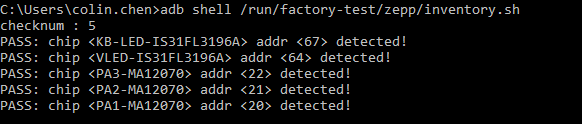
i2c-3 64 VLED-IS31FL3196A

i2c-3 67 KB-LED-IS31FL3196A

Test command :

adb shell /etc/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 :

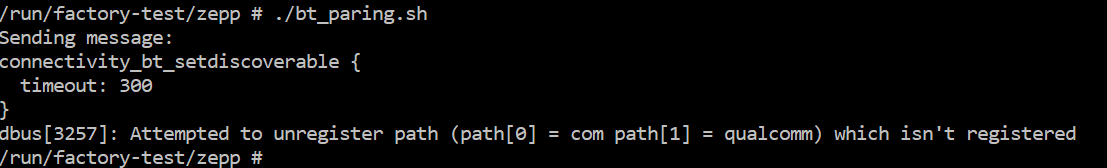
adb shell /etc/factory-test/zepp/get\_pa\_fault.sh

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-alexa1 avs\_led3

led-alexa2 avs\_led4

led-formation1 key\_led1

led-formation2 key\_led2

led-formation3 key\_led3

led-foot1 ft\_led1

led-foot2 ft\_led2

led-foot3 ft\_led3

led-foot4 ft\_led4

led-foot5 ft\_led5

led-foot6 ft\_led6

led-foot7 ft\_led7

led-foot8 ft\_led8

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

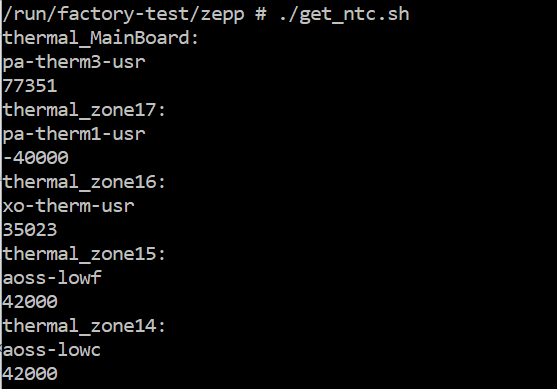
output form:

<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:

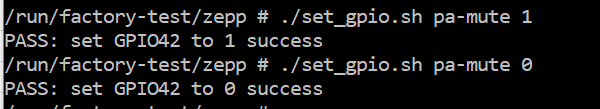
pa-mute 42

pa-en 40

Command form:

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

Exp:

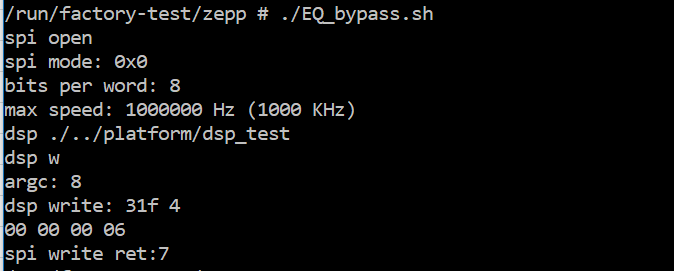


## 2.8 Audio set

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”

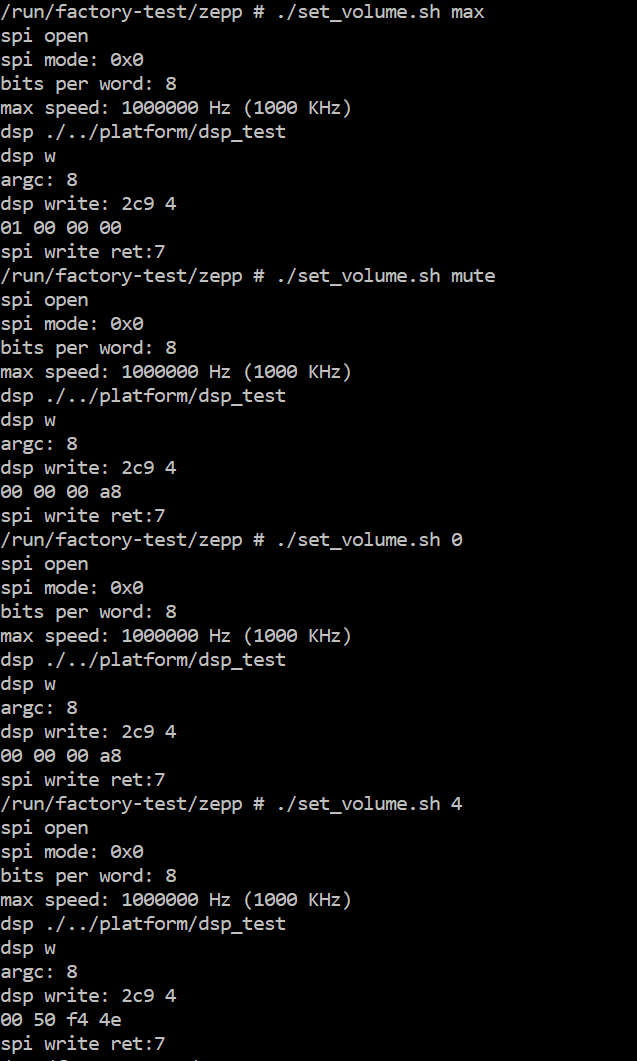
Volume detail data define at volume.conf from 1st line to last line was mute volume to max volume

Command form:

./set\_volumen.sh <value>

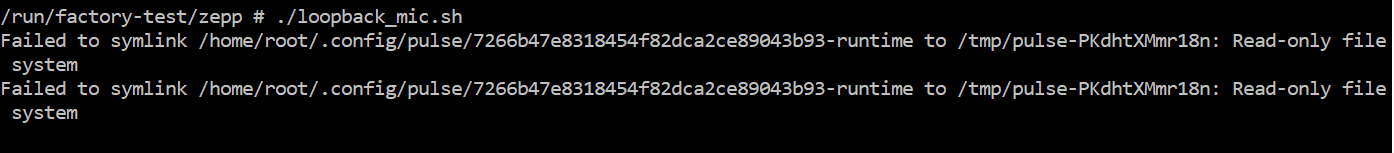
Value support number from 0~max volume level also “mute”, “max”

Exp:

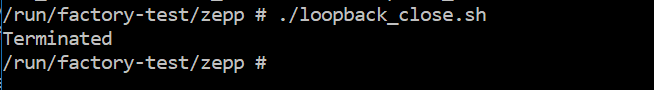


## 2.10 Mic loopback

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



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



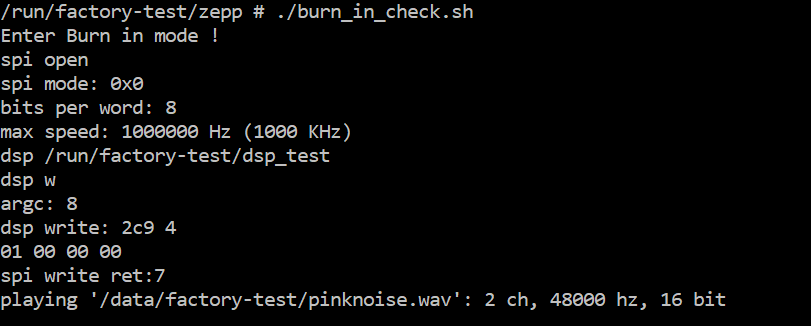
## 2.11 burn in mode

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

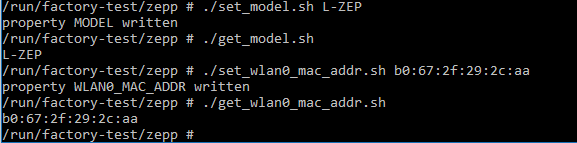
|  |  |
| --- | --- |
| **Script** | **Function** |
| nvram\_fs.sh | Formal NVRAM, after formal need to reboot to active it. |
| get\_assembly.sh | Get ASSEMBLY No. |
| Set\_assembly.sh | Set ASSEMBLY No. |
| get\_board\_version.sh | Get board version |
| get\_country\_code.sh | Get country code |
| 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:



## 2.14 Set Wlan0 MAC address

Formal:

./set\_wlan0\_mac\_addr.sh <6 bytes hex number>

Exp: ./set\_wlan0\_mac\_addr.sh 000AF5898FF

And the get command was:

./get\_wlan0\_mac\_addr.sh

## 2.15 Set BT MAC address

Forml:

./set\_bt0\_mac\_addr.sh <6 bytes hex number with ‘:’ split >

Exp:

./set\_bt0\_mac\_addr.sh 70:c9:4e:9b:64:11

And the get command was

./get\_bt0\_mac\_addr.sh

## 2.15 Mic record

Formal:

./mic\_record.sh <record time in seconds>

Exp:

./mic\_record.sh 4

When running this script will need 11S for 4 channels mic audio path setting. Then will record audio to “/data/factory-test/mic-4ch.wav”

## 2.16 Mic record close

Formal:

./mic\_record\_close.sh <number of record time in second>

Exp:

./mic\_record\_close.sh 4

Means close record after recording 4 seconds, please notice: Mic record must run just before mic record close.

Exp:

adb shell /etc/factory-test/zepp/mic\_record.sh

adb shell /etc/factory-test/zepp/mic\_record\_close.sh 4

Means: start audio record and record audio for 4 seconds then close it.

## 2.17 Audio play

Formal:

./audio\_play.sh <audio path>

Exp:

./audio play.sh /data/pinknoisc.wav

## 2.18 Audio play close

Formal :

./audio\_play\_close.sh

## 2.19 Check audio ready

Formal:

./check\_audio\_ready.sh

When running this script, will detect the audio serve and auto end the script when audio serve ready.

## 2.20 Enter Wifi/BT FTM mode

Formal:

nohup “./en\_ftm.sh > /dev/null” &

exp:

adb shell nohup “/etc/factory-test/zepp/en\_ftm.sh > /dev/null” &

## 2.21 exit Wifi/BT FTM mode

Formal:

./dis\_ftm.sh

## 2.22 SPDIF On

Formal :

nohup “./spdif\_on.sh > /dev/null” &

exp:

adb shell nohup “/etc/factory-test/zepp/spdif\_on.sh > /dev/null” &

notice: spdif was pipe play, need to hold at background.

## 2.23 SPDIF OFF

Formal:

./spdif\_off.sh