RTL8723DE
RTL8723DU
RTL8723DS
COB
MP FLOW
R04

Table of Contents

List of Tables

List of Figures

Revision History:

Revision	Date	Changes	Author
R01	2017/07/03	First Release	*Winnita.Zhao*
R02	2017/07/26	Modify 2.3.1	*Winnita.Zhao*
R03	2017/07/31	Add Chap6(K-FREE FLOW)	*Winnita.Zhao*
R04	2017/10/20	Reformats	*Winnita.Zhao*

1. Test Platform

The calibration flow described in following section is based on WiFi tester. The tester qualified by Realtek is listed below:

Include 802.11ac Test		Only 802.11a/b/g/n Test	
Vendor	**Modal Name**	**Vendor**	**Modal Name**
LitePoint	IQxel	LitePoint	IQFlex
Itest	WT-200	LitePoint	IQView
NI	PXIe-5644R/5645R	LitePoint	IQnxn
Anritsu	MT8870A	Agilent	N4010A
Aeroflex	PXI 3000 Series		

The test environment setup is as below:

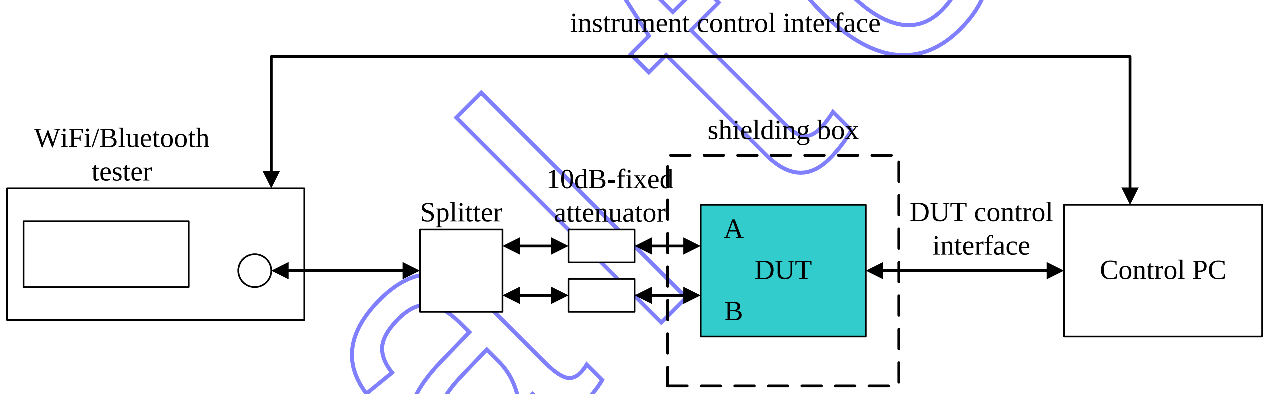

Figure 1: Brief diagram about test environment setup of DUT

Note the 10dB-fixed attenuator has to set as close as possible to DUT since it will reduce the mismatch effect between DUT and the environment.

2. DUT MP Flow

Below diagram shows a global view of mass production flow, please refer to following sub-section to get detailed description for each step.

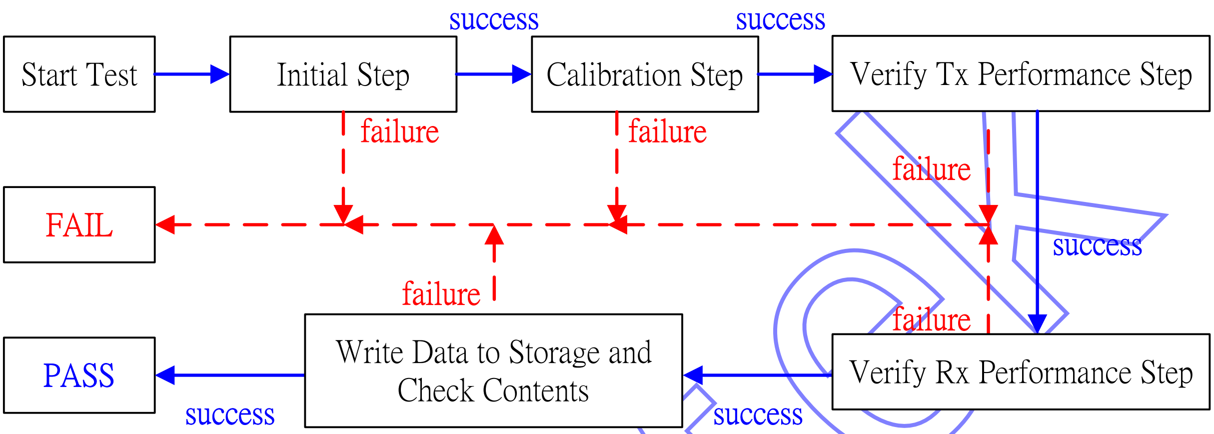

Figure 2: Brief diagram about MP flow

2.1. Environment Setup

Step1:

Use USB connect PC and Android/Linux device, as shown Figure 3.

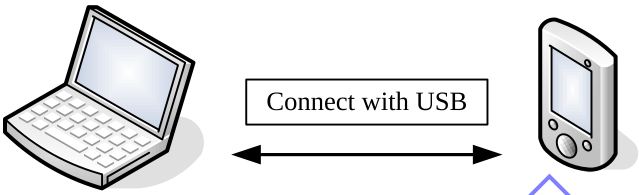

Figure 3: Connect PC to Android/Linux device

Step2:

Use adb command as below:

adb devices

to check Android/Linux device to connect or not. If Android/Linux device connect successfully, screen show as Figure 4.

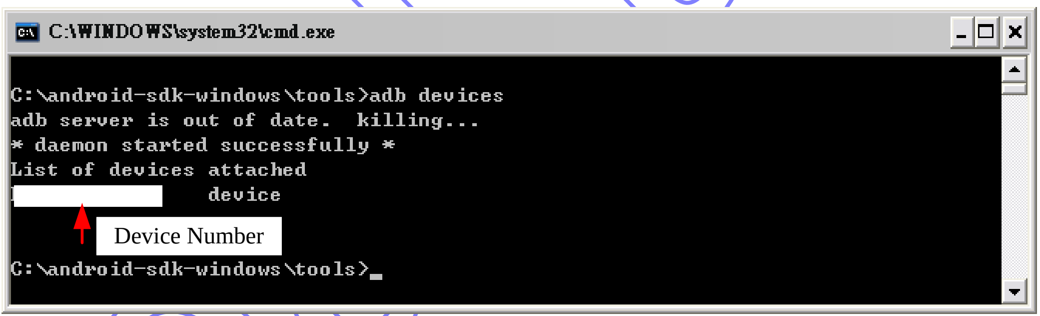

Figure 4: Check computer connect to Android/Linux device

Step3:

Use adb command as below:

adb shell

to enter Android/Linux device command line, as shown Figure 5.

Figure 5: Enter Android/Linux device command line

2.2. WiFi Initial Step

The initial WiFi step as shown in Figure 6:

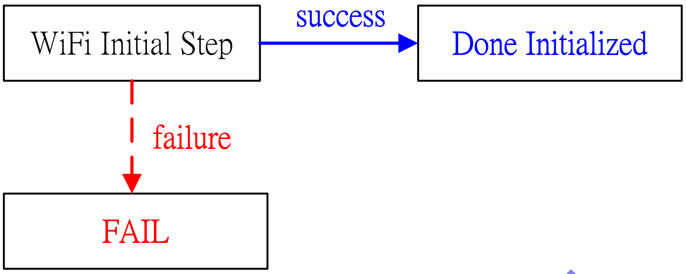

Figure 6: Brief diagram about initial WiFi step

2.2.1. Initial WiFi

Refer to Section 2.1, you can connect PC and Android/Linux device,

The relative control commands about initializing WiFi DUT. The main command is

Android use rtwpriv command. Linux use rtwpriv command.

EX: Android system: rtwpriv wlan0 mp_start

Linux system : rtwpriv wlan0 mp_start

The following command is an example for Linux

remount

root

rmmod wlan

insmod wlan.ko

(delay 5 second)

ifconfig wlan0 up

rtwpriv wlan0 mp_start

2.3. WiFi Calibration Step

This step includes 2 sub-steps as shown in Figure 7:

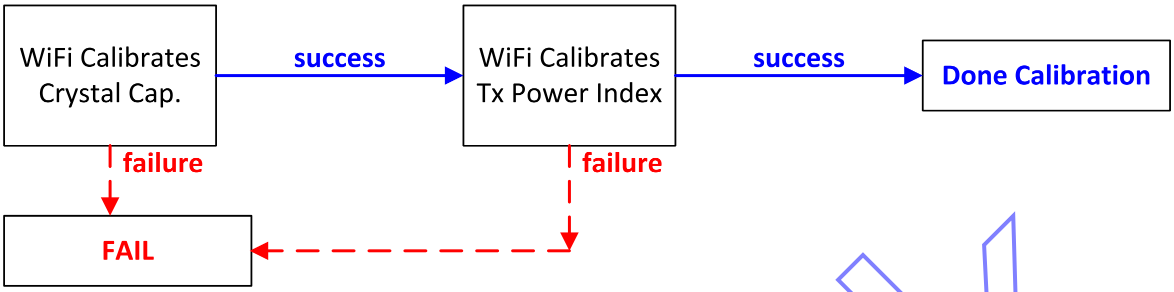

Figure 7: Brief diagram about WiFi calibration step

2.3.1. WiFi Calibrates Crystal Cap.

2.3.1.1. EFuse definition about Crystal Cap.

First, take a look at eFuse content about setting of Crystal Cap. . Normal driver will load this value in initial step. So this value must be well-calibrated and filled on correct eFuse location.

Crystal Calibration	0xB9[5:0]

Table 1: Crystal calibration offset in eFuse

2.3.1.2. Calibrated Crystal Cap. Flow

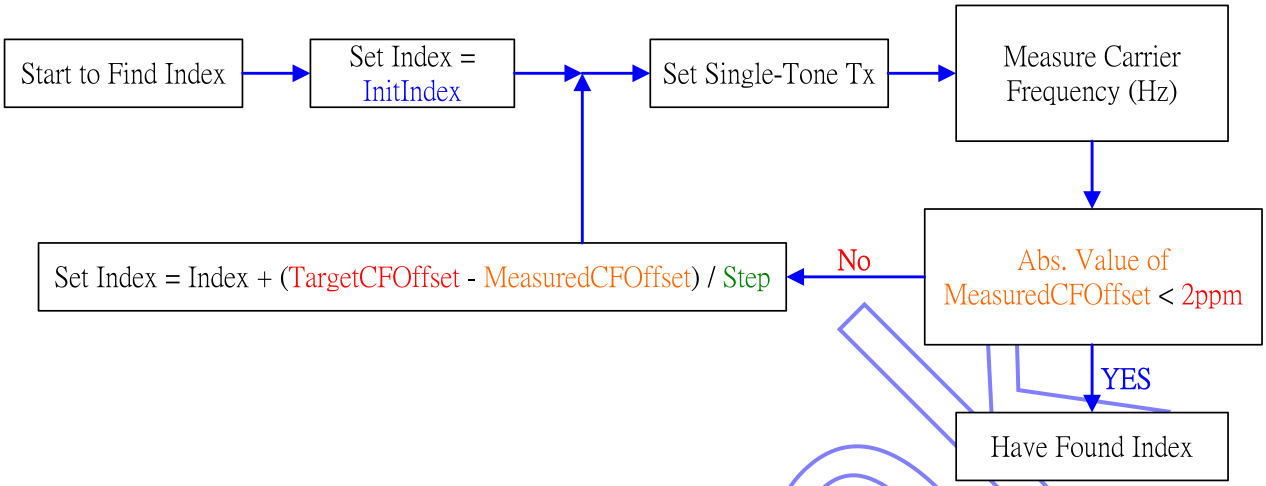

Figure 8: Finding Crystal Cap. Index Flow

InitIndex: the default value is 0x20. Index range is 0x0 to 0x3F.

MeasuredCFOffset: Carrier frequency measured by instrument - Ideal Carrier Frequency
Target range Abs. Value of 2ppm in 2.4GHz band is about 10KHz(±5KHz).

TargetCFOffset: generally is 0 ppm

Step: This value is dependent of different module dominated by external capacitor beside the crystal, so it needs to modify easily in initial file of test program. Usually, the value is about +2 ~ +3KHz by experience. The suitable value should be checked by Hardware RD and fill it in the setup file of test program. The plus symbol means that the crystal cap. index and carrier frequency is positive-dependent (The larger index is relative to major carrier frequency).

Single tone command is as below:

Step 1:

rtwpriv wlan0 mp_ant_tx a
rtwpriv wlan0 mp_channel 7
rtwpriv wlan0 mp_txpower patha=0,pathb=0
rtwpriv wlan0 mp_phypara xcap=32
rtwpriv wlan0 mp_ctx background,stone

Step 2:

You can measure Frequency Error(ppm) Freq_Err. If Freq_Erris is between ±2ppm，you found this value (index_cry_ok) correctly. If Freq_Erris is not between ±2ppm, you caught use algorithm as below to find next index_cry_next until you find Freq_Err between ±2ppm.

$$\text{index\_cry\_next} = \text{Index\_cry} - \frac{Freq\_Err \times 2442}{2500}$$

ex: We set Index_cry =32, measure Freq_Err = -23.49ppm,

$$\text{The next index is index\_cry\_next} = 32 - \frac{(-23.49) \times 2442}{2500} \doteq 55$$

step3: The index_cry_next must be **rounded the number to the integer**, and use command to update index as follows:

rtwpriv wlan0 mp_phypara xcap index_cry_next

ex: rtwpriv wlan0 mp_phypara xcap 55

Repeat Step2, Step3 until find Freq_Erris correctly and note down Index_cry_ok.

Stop sing tone command is as below:

rtwpriv wlan0 mp_ctx stop

2.3.2. WiFi Calibrates Tx Power Index

2.3.2.1. EFuse definition about Tx power index and thermal meter

First, take a look at eFuse content about setting of RF Tx gain index. Normal driver will load bellow Tx gain setting for each channel group or each PHY data rate. So these Tx gain setting must be well-calibrated and filled on correct eFuse location.

Power Index Location in EFuse of Antenna S1						
	Group 1 CH1 – CH2	Group 2 CH3 – CH5	Group 3 CH6 – CH8	Group 4 CH9 – CH11	Group 5 CH12 – CH13	Group 6 CH14
MCS7 B40	0x16[7:0]	0x17[7:0]	0x18[7:0]	0x19[7:0]	0x1A[7:0]	
CCK	0x10[7:0]	0x11[7:0]	0x12[7:0]	0x13[7:0]	0x14[7:0]	0x15[7:0]

Power Index Location in EFuse of Antenna S0						
	Group 1 CH1 – CH2	Group 2 CH3 – CH5	Group 3 CH6 – CH8	Group 4 CH9 – CH11	Group 5 CH12 – CH13	Group 6 CH14
MCS7 B40	0x40[7:0]	0x41[7:0]	0x42[7:0]	0x43[7:0]	0x44[7:0]	
CCK	0x3A[7:0]	0x3B[7:0]	0x3C[7:0]	0x3D[7:0]	0x3E[7:0]	0x3F[7:0]

Power Difference Location in EFuse of Antenna S1		
2G Band	54M-1T to MCS7-B40	0x1B[3:0]
	MCS7-B20 to MCS7-B40	0x1B[7:4]

Power Difference Location in EFuse of Antenna S0		
2G Band	54M-1T to MCS7-B40	0x45[3:0]
	MCS7-B20 to MCS7-B40	0x45[7:4]

Thermal Meter	0xBA[7:0]

Table 2: Tx gain index and thermal meter offset in eFuse

2.3.2.2. Define target power

According to
EMI/EMC regulatory
IEEE TX EVM / Spectrum Mask requirement
Then you can define your target power for each channel group and also each PHY data rate. The recommended target power is listed below and assumes all channel have the same target power for each PHY data rate.

Data Rate	MCS7-B40	MCS7-B20	54M	CCK
Target Power 2G	13dBm	13dBm	14dBm	16dBm

Table 3: The default target power

2.3.2.3. Tx calibration flow

Theoretically, we need to measure all value defined above in eFuse to calibrate the Tx power level. But since it needs too much time, we only measure several channels with MCS7-B40 signal and figure out the other non-measured value by some easy mathematics method.

Usually, the recommended measured channels are listed below:

2G Band	
CH4	CH10

Table 4: The recommended measured channel for Tx power calibration

The flow is shown as below:

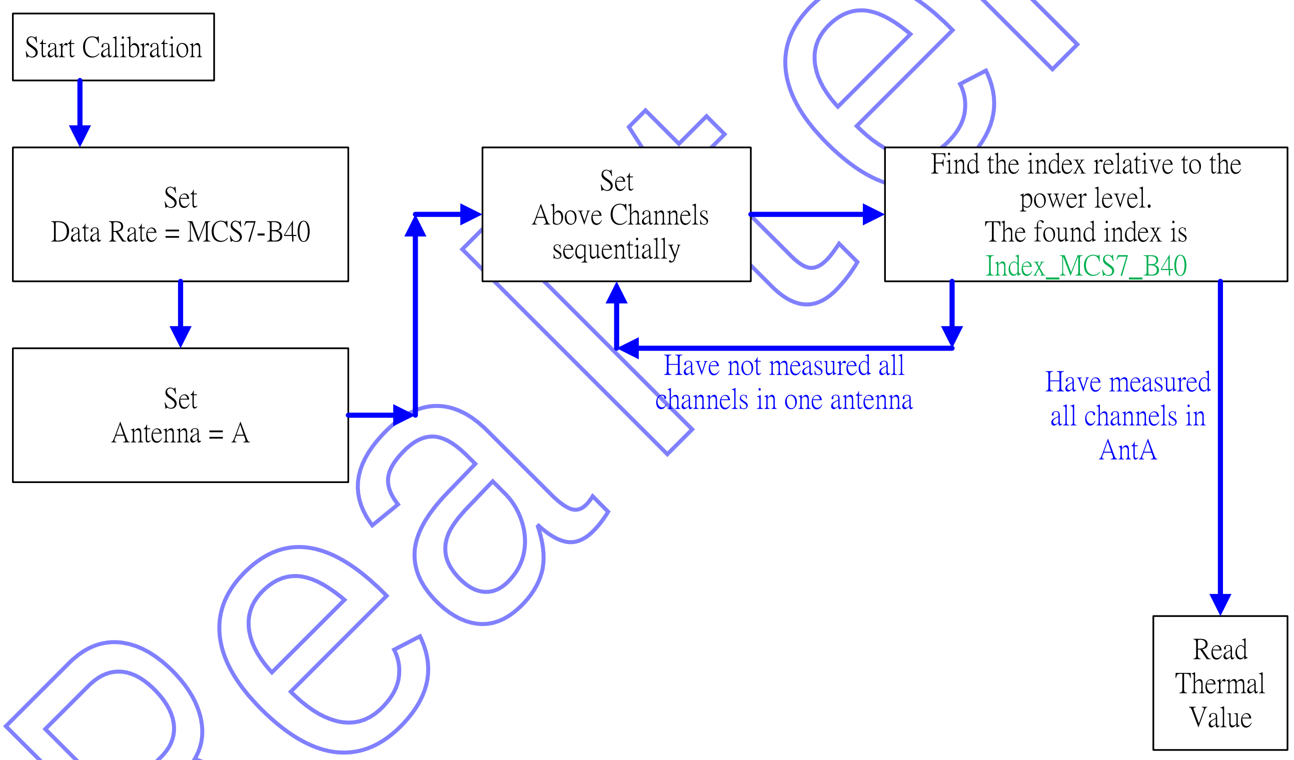

Figure 9: Tx calibration flow

Each finding index flow is shown as below:

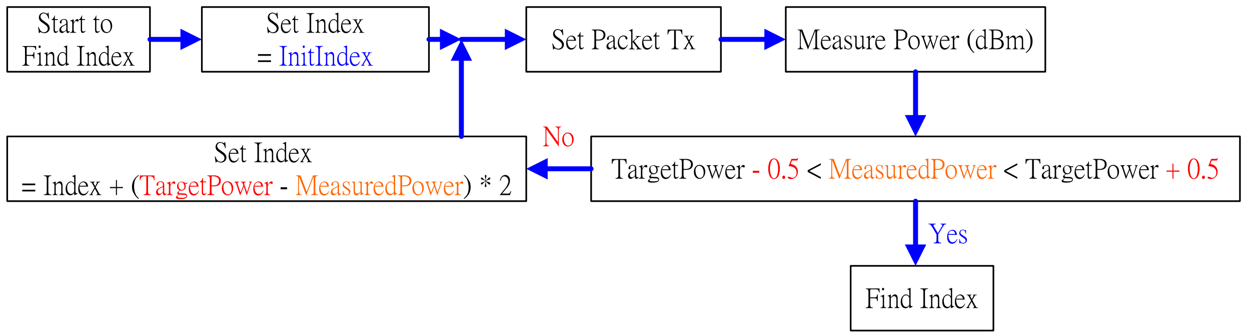

Figure 10: Finding index flow

While finding Index_MCS7_B40, the Init Index is defined by user or programmer and target power is MCS7-B40 target power level defined before.

After finding all Index_MCS7_B40, use these values to get all Tx gain index in each channel set by interpolation. As 2G channels for example, if the measured index in CH4 is 41 and the measured index in CH10 is 43, all 2G group MCS7-B40 index is shown as below:

Group 1	Group 2	Group 3	Group 4	Group 5
40 (Calculated by Interpolation)	41 (Measured)	42 (Calculated by Interpolation)	43 (Measured)	44 (Calculated by Interpolation)

Table 5: The example of finding index in 2G band by interpolation

We have measured all groups of 5G band 1 and 5G band 2, so just only use the interpolation for 5G band 3 and 5G band4.
The 2G CCK index will be a fix offset to MCS7-B40 dependent on CCK and MCS7-B40 target power difference and input baseband signal amplitude. So it also needs to be checked by RD and fill it to the setup file of test program. If we define a CCK offset value as CCK_Offset, the CCK index in above measured example will be shown as below:

Group 1	Group 2	Group 3	Group 4	Group 5	Group 6
40 + CCK_Offset	41 + CCK_Offset	42 + CCK_Offset	43 + CCK_Offset	44 + CCK_Offset	45 + CCK_Offset

Table 6: The example of finding CCK index

The all power difference values are between +7 and -8. The value 0x0 ~ 0x7 in eFuse means 0 ~ +7 and the value 0x8 ~ 0xF in eFuse means -8 ~ -1. The +1 power difference will plus 0.5dBm power theoretically, so we calculate all power difference by the defined target power. We take Table 7 as an example, the 2G power difference is shown as below:

MCS7-B20 to MCS7-B40	(MCS7-B20_Tatget_Power - MCS7-B40_Tatget_Power)x2 = ((13 – 13) x2) = 0
54M-1T to MCS7-B40	(54M_Tatget_Power - MCS7-B40_Tatget_Power)x2 = ((14 – 13) x2) = 2

Table 7: The example of finding power difference

For example:

Step 1:

You must offset cable loss by yourself.

Ex: Rate **MCS7**, Bandwidth**40**, Channel **4** , Tx ant =S1, Tx_target = 13dBm, Tx power index=42, find **Index_MCS7_B40_Group2**

The command is shown as below:

```
rtwpriv wlan0 mp_ant_tx a                         // set ant S1(S1:a; S0:b)
rtwpriv wlan0 mp_bandwidth 40M=1,shortGI=0 //set bandwidth 40M, short GI off
rtwpriv wlan0 mp_channel 4                           //set Channel 4
rtwpriv wlan0 mp_rate 135                         //set OFDM data rate MCS7
rtwpriv wlan0 mp_txpower patha=42,pathb=0    //set Tx power index 42
                                        (For 8723D, no matter S1 or S0, use patha)
rtwpriv wlan0 mp_ctx background,pkt                //start continuous Tx
```

After finish measuring, use command as below to stop Tx.

```
rtwpriv wlan0 mp_ctx stop                    //stop continuous Tx
```

Step 2:

After you finish measuring, you can get power. If power is between (13 dBm±0.5 dBm)，you found this power index(**Index_MCS7_B40_Group2=**42) correctly. If power is not between (13dBm±0.5 dBm) you caught use algorithm as below to find next **Index_MCS7_B40_Group2** until power is between (13 dBm±0.5 dBm).

$$\text{Index\_MCS7\_B40\_ Group}\mathbf{2}\text{\_next} = \text{Index\_MCS7\_B40\_ Group}\mathbf{2}+\frac{(Tx\_target - Tx\_measure)}{0.5}$$

ex: Measuring power is 11.6 dBm

Index_MCS7_B40_ Group**2**_next = 42 + $\frac{(13 - 11.6)}{0.5}$ ≒ **45** (The value must be **rounded the number to the integer**)

Rate **MCS7**, Bandwidth**40**, Channel **4** , Tx_target = 13dBm, Tx power index=45, find **Index_MCS7_B40_Group2**

rtwpriv wlan0 mp_bandwidth 40M=1,shortGI=0 //set bandwidth 40M, short GI off

rtwpriv wlan0 mp_channel 4 //set Channel 4

rtwpriv wlan0 mp_rate 135 //set rate MCS7

rtwpriv wlan0 mp_txpower patha=45,pathb=0 //set Tx power index 42, ant A

rtwpriv wlan0 mp_ctx background,pkt //start continuous Tx

After finish measuring, use command as below to stop Tx.

rtwpriv wlan0 mp_ctx stop //stop continuous Tx

Repeat Step1, Step2 until find measuring power correctly and note down **Index_MCS7_B40_Group2**.

2.3.2.4. Read Thermal Meter

Normal driver will load thermal meter to do power tracking. So this value must be filled on correct eFuse location. Use MP API function below to get thermal meter value:

rtwpriv wlan0 mp_ther

You get value response to Android/Linux device, and note down thermal meter value.

ex: wlan0 mp_ther:14

2.4. Bluetooth Calibrates Tx Power Index by Non-Signaling mode

2.4.1. Efuse definition about Tx power index

First, take a look at eFuse (config file) content about setting of power index and channel adjust value . Normal driver will load this value in initial step. So this value must be well-calibrated and filled on correct eFuse location

Efuse Offset	Explanation
0x15A[7:0]	ALL MAX Power Index
0x15B[7:0]	1M Default Power Index
0x15C[7:0]	2M Default Power Index
0x15D[7:0]	3M Default Power Index
0x15E[7:0]	LE Default Power Index
0x15F[7:0]	0x01

Table 8: Tx power index offset in eFuse

2.4.2. Tx Calibration Flow

Usually, the recommended measured channels are listed below:

Test Channel		Packet Type	Bluetooth Spec.
Ch 6	2408MHz	DH1	> 0 dBm

Table 9: The recommended measured channel for Tx power calibration

Calibration output Tx power index flow is as below

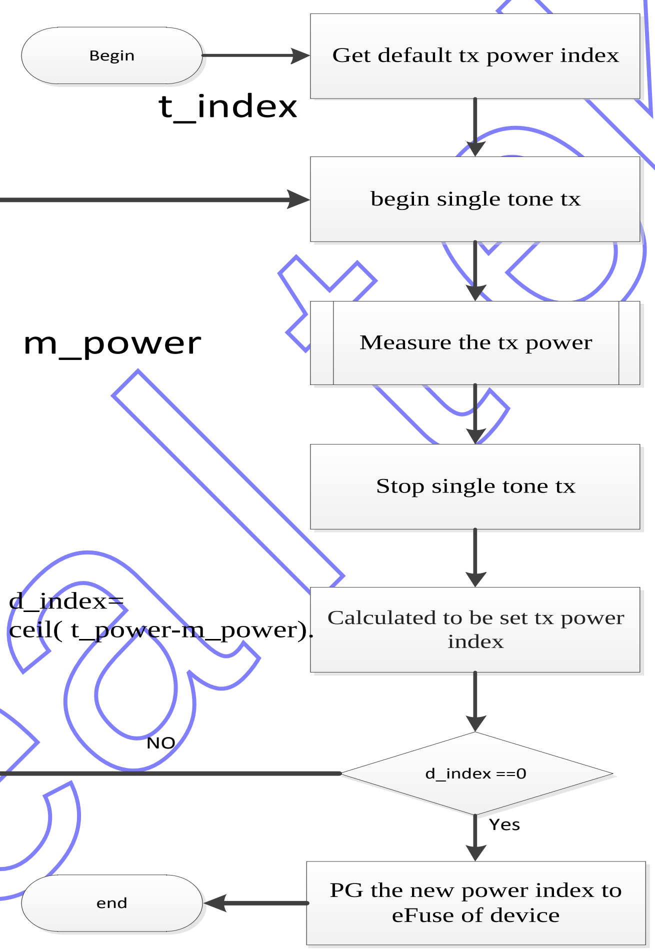

Figure 11: Calibration output tx power index flow

t_index: The default tx power index . It can be known by the command of mp tool.

m_power: Measured tx power.

t_power: Target tx power, ex. 4.5dbm

d_index: Define d_index = floor(t_power - m_power).

EX: if t_power - m_power = 0.2 , d_index = 0;

if t_power - m_power = -0.5, d_index =-1;

if t_power - m_power = -1.2, d_index=-2;

TX output power calibration step by step index is defined below:

First , target Tx power is t_power .The target Tx power index can be defined by the customer .

Step1): Get default tx power index, the index is t_index. Customers can get the default value by the HCI command. Please refer to Section 3.3.3

Step2): Begin single tone tx in channel 0(2402MHz).

Step3): Measure the Tx power , the Tx power is m_power. Then to stop single tone Tx.

Step4): Calculated the Tx power index gap.
d_index = floor(t_power - m_power).

Step5): To set new tx power index . The new index t_index defined as:
t_index = t_index + d_index;
8723D Bluetooth Power Index Step is 1.0 dBm.

Step6): PG the new power index t_index to eFuse of device.

Step7): Re-begin single tone tx in channel 0. (Double confirm the Tx power)

Step8): Measure the Tx power

Step9): Stop single tone tx

EX:

Note: The chip is RTL8723DS . The power index step is 1.0 dBm.

The Target tx power t_power is 4.5 dbm and the default tx power index t_index is 26 .

t_power = 4.5 dbm

t_index = 26.

Step1) Measure the tx power

Begin single tone tx →> Measure the tx power m_power = 3.4 dbm →stop single tone tx

Then d_index = floor(4.5 - 3.4) = floor(1.1) = 1

Step2) Calculate new tx power index

t_index = t_index + d_index = 26 + 1 = 27 .

Step3) Write the new tx power index 27 to efuse or config file.

Step4) verify Tx power .

Begin single tone tx → Measure the tx power m_power = 4.4 dbm → stop single tone tx

3. TRx Verify Performance Step

3.1. Verify WiFi Performance Step

3.1.1. Verify WiFi Tx Performance

Use the calibrated index in previous step and measure Tx power, EVM, frequency offset and LO leakage to check Tx performance is ok or not. The recommended test items are listed below:

Data Rate	Antenna	Channel	Item	Criterion
MCS7-B40	Antenna S1 Antenna S0	CH10	Power	Typical: 13dBm, Acceptable Range: +1/-1.5dB
			EVM	< -28dB
			Freq. Err.	±15ppm
			Leakage	< -20dBtotal
			Mask	IEEE spec. defined
MCS7-B20	Antenna S1 Antenna S0	CH10	Power	Typical: 13dBm, Acceptable Range: +1/-1.5dB
			EVM	< -27dB
			Freq. Err.	±15ppm
			Leakage	< -20dBtotal
			Mask	IEEE spec. defined
OFDM 54M	Antenna S1 Antenna S0	CH1	Power	Typical: 14dBm, Acceptable Range: +1/-1.5dB
			EVM	< -25dB
			Freq. Err.	±15ppm
			Leakage	< -15dBtotal
			Mask	IEEE spec. defined
CCK 11M	Antenna S1 Antenna S0	CH7	Power	Typical: 16dBm, Acceptable Range: +1/-1.5dB
			EVM	< 8%
			Freq. Err.	±15ppm
			Mask	IEEE spec. defined

Table 10: The recommended test items of WiFi Tx

Please refer to Section 2.3.2.3, you can calculate the following index information:

Index_MCS7_B20_Group1 = Index_MCS7_B40_Group1 + (MCS7-B20 to MCS7-B40)
Index_MCS7_B20_Group2 = Index_MCS7_B40_Group2 + (MCS7-B20 to MCS7-B40)
Index_MCS7_B20_Group3 = Index_MCS7_B40_Group3 + (MCS7-B20 to MCS7-B40)
Index_MCS7_B20_Group4 = Index_MCS7_B40_Group4 + (MCS7-B20 to MCS7-B40)
Index_MCS7_B20_Group5 = Index_MCS7_B40_Group5 + (MCS7-B20 to MCS7-B40)

Index_OFDM_Group1= Index_MCS7_B40_Group1 + (54M-1T to MCS7-B40)
Index_OFDM_Group2= Index_MCS7_B40_Group2 + (54M-1T to MCS7-B40)
Index_OFDM_Group3= Index_MCS7_B40_Group3 + (54M-1T to MCS7-B40)
Index_OFDM_Group4= Index_MCS7_B40_Group4 + (54M-1T to MCS7-B40)
Index_OFDM_Group5= Index_MCS7_B40_Group5 + (54M-1T to MCS7-B40)

For example:
You want to verify WiFi Tx Performance and you can follow step as below.

Step 1: Initial WiFi Step
Please refer to Section 2.2.1

Step 2: start Tx
Ex: Rate **MCS7**, Bandwidth**40**, Channel **4** , Tx_target = 13dBm, **Index_MCS7_B40_Group2 = 42(the value is from map)**

rtwpriv wlan0 mp_bandwidth 40M=1,shortGI=0
rtwpriv wlan0 mp_channel 4
rtwpriv wlan0 mp_rate 135
rtwpriv wlan0 mp_txpower patha=42,pathb=0
rtwpriv wlan0 mp_ctx background,pkt

After finish measuring, use command as below to stop Tx.
rtwpriv wlan0 mp_ctx stop **//stop continuous Tx**

Ex: Rate **MCS7**, Bandwidth**20**, Channel **4** , Tx_target = 13dBm, **Index_MCS7_B20_Group2 = 42(the value is from map** (Index_MCS7_B20_Group2 = Index_MCS7_B40_Group2 + (MCS7-B20 to MCS7-B40))

rtwpriv wlan0 mp_bandwidth 40M=0,shortGI=0

rtwpriv wlan0 mp_channel 4
rtwpriv wlan0 mp_rate 135
rtwpriv wlan0 mp_txpower patha=42,pathb=0
rtwpriv wlan0 mp_ctx background,pkt

After finish measuring, use command as below to stop Tx.

rtwpriv wlan0 mp_ctx stop **//stop continuous Tx**

Ex: Rate **OFDM54**, Bandwidth**20**, Channel **10** , Tx_target = 13dBm,
Index_OFDM_B20_Group4 = 42(the value is from map(Index_OFDM_Group4= Index_MCS7_B40_Group4 + (54M-1T to MCS7-B40))

rtwpriv wlan0 mp_bandwidth 40M=0,shortGI=0
rtwpriv wlan0 mp_channel 10
rtwpriv wlan0 mp_rate 108
rtwpriv wlan0 mp_txpower patha=42,pathb=0
rtwpriv wlan0 mp_ctx background,pkt

After finish measuring, use command as below to stop Tx.

rtwpriv wlan0 mp_ctx stop **//stop continuous Tx**

PS: Data Rate ID:
[CCK: 1 M 2 M 5.5 M 11M] x 2 = 2 4 11 22
[OFDM: 6M 9M 12 M 18 M 24 M 36 M 48 M 54M] x 2 = 12 18 24 36 48 72 96 108
[HT 1S MCS0 ~ MCS7] : [MCS0]=128, [MCS1]=129, [MCS1]=130 ~ [MCS7]=135

Bandwidth command
rtwpriv wlan0 mp_bandwidth 40M=1,shortGI=0 //set bandwidth 40M
rtwpriv wlan0 mp_bandwidth 40M=0,shortGI=0 //set bandwidth 20M

3.1.2. Verify WiFi Rx Performance

Measure the DUT Rx sensitivity to check Rx performance is ok or not. The recommended test items are listed below:

Data Rate	Antenna	Channel	Item	Criterion
MCS7-B40	Antenna S1 Antenna S0	CH10	Sensitivity	< -64dBm (1R)
MCS7-B20	Antenna S1 Antenna S0	CH10		< -67dBm (1R)
OFDM 54M	Antenna S1 Antenna S0	CH1		< -71dBm (1R)
CCK 11M	Antenna S1 Antenna S0	CH7		< -83dBm (1R)

Table 11: The recommended test items of WiFi Rx

For example:
You want to verify WiFi Tx Performance and you can follow step as below.

Step 1: Initial WiFi Step
Please refer to Section 2.2.1.

Step 2: Verify Rx

ex: Start Rx Test(use 802.11b, channel 1, **11**Mbps, Antenna S1, Bandwidth 20)

rtwpriv wlan0 mp_ant_rx a
rtwpriv wlan0 mp_bandwidth 40M=0,shortGI=0
rtwpriv wlan0 mp_channel 1
rtwpriv wlan0 mp_arx start
rtwpriv wlan0 mp_reset_stats

Step 3: Get report

rtwpriv wlan0 mp_arx mac

Step 4: Stop Rx Test

rtwpriv wlan0 mp_arx stop

3.2. Verify Bluetooth Performance Step

3.2.1. Verify Bluetooth Tx Performance

BT MAP is burned in EFuse already. Only verify Bluetooth TX/RX performance.

Bluetooth Tx criterion is shown as below:

	Test Item	Sub Test Item	Packet Type	Channel	Criterion Bluetooth Spec.
Verify Tx	Maximum Output Power	Average Power	DH1	Low (CH6)	> 0dBm
				Middle (CH42)	> 0dBm
				High (CH70)	> 0dBm
	Modulation Characteristics	Delta F1 Avg.	DH1	All	140KHz ~ 175KHz
		Delta F2 Avg.			140KHz ~ 175KHz
		Delta F2 Max.			> 115KHz
		Modulation Index			> 0.8
	Initial Carrier Frequency Error		DH1	All	-20KHz ~ 20KHz

Table 12: The recommended test items of Bluetooth Tx

EX:

Step 1: Enter MP Mode and download patch code

```
root@android:/ # rtlbtmp
rtlbtmp

::::::::::::::::::::::::::::::::::::::::::::::::
:::::::: Bluetooth MP Test Tool Starting ::::::::
> enable uart:/dev/ttyS0
enable uart:/dev/ttyS0
> > > enable[Success:0]
```

Step 2: Set default table

```
> bt_mp_Exec 5
bt_mp_Exec 5
bt_mp_Exec[Success:0]
> bt_mp_Exec,5,0x00


> bt_mp_Exec 6
bt_mp_Exec 6
bt_mp_Exec[Success:0]
> bt_mp_Exec,6,0x00
```

Step 3: Set Parameter :

You can use “bt_mp_SetParam” to set parameters and can use ”bt_mp_GetParam” to check it.
The Format is

bt_mp_SetParam Index0,value0; Index1,..;IndexN,valueN

Example : If you want to set the channel 10 and packet type “BT_PKT_3DH5”, you can use”

bt_mp_SetParam 0x01,0x0a;0x02,0x08”

Test Item		adb command
	Test Item	Channel = 6
DH1	Maximum Power	bt_mp_SetParam 1,0x06;2,0x00;3,0x07;4,0x00;6,0x7F;7,0x7;9,0x17F0E;11,0x0000009e8b33
DH1	Delta F1	bt_mp_SetParam 1,0x06;2,0x00;3,0x05;4,0x00;6,0xFF;7,0x7;9,0x17F0E;11,0x0000009e8b33
DH1	Delta F2	bt_mp_SetParam 1,0x06;2,0x00;3,0x02;4,0x00;6,0xFF;7,0x7;9,0x17F0E;11,0x0000009e8b33
3DH1	ALL	bt_mp_SetParam 1,0x06;2,0x06;3,0x07;4,0x00;6,0x7F;7,0x7;9,0x31B6E;11,0x0000009e8b33

Test Item		adb command
	Test Item	Channel = 42
DH1	Maximum Power	bt_mp_SetParam 1,0x2a;2,0x00;3,0x07;4,0x00;6,0x7F;7,0x7;9,0x17F0E;11,0x0000009e8b33
DH1	Delta F1	bt_mp_SetParam 1,0x2a;2,0x00;3,0x05;4,0x00;6,0xFF;7,0x7;9,0x17F0E;11,0x0000009e8b33
DH1	Delta F2	bt_mp_SetParam 1,0x2a;2,0x00;3,0x02;4,0x00;6,0xFF;7,0x7;9,0x17F0E;11,0x0000009e8b33
3DH1	ALL	bt_mp_SetParam 1,0x2a;2,0x06;3,0x07;4,0x00;6,0x7F;7,0x7;9,0x31B6E;11,0x0000009e8b33

Test Item		adb command
	Test Item	Channel = 70
DH1	Maximum Power	bt_mp_SetParam 1,0x46;2,0x00;3,0x07;4,0x00;6,0x7F;7,0x7;9,0x17F0E;11,0x0000009e8b33
DH1	Delta F1	bt_mp_SetParam 1,0x46;2,0x00;3,0x05;4,0x00;6,0xFF;7,0x7;9,0x17F0E;11,0x0000009e8b33
DH1	Delta F2	bt_mp_SetParam 1,0x46;2,0x00;3,0x02;4,0x00;6,0xFF;7,0x7;9,0x17F0E;11,0x0000009e8b33
3DH1	ALL	bt_mp_SetParam 1,0x46;2,0x06;3,0x07;4,0x00;6,0x7F;7,0x7;9,0x31B6E;11,0x0000009e8b33

Step 4: Run Packe Tx

bt_mp_Exec 12

Step 5: measured by Bluetooth test instrument(ex. Letepoint IQNxN)

bt_mp_Report 1

bt_mp_Report 1” should be executed every 1s

COMMAND	INDEX
HCI_RESET	0
TEST_MODE_ENABLE	1
PG_EFUSE_RAWDATA	2
SET_TX_GAIN_TABLE	3
SET_TX_DAC_TABLE	4
SET_DEFAULT_TX_GAIN_TABLE	5
SET_DEFAULT_TX_DAC_TABLE	6
SET_POWER_GAIN_INDEX	7
SET_POWER_GAIN	8
SET_POWER_DAC	9
SET_XTAL	10
REPORT_CLEAR	11
PACKET_TX_START	12
PACKET_TX_UPDATE	13
PACKET_TX_STOP	14
CONTINUE_TX_START	15
CONTINUE_TX_UPDATE	16
CONTINUE_TX_STOP	17
PACKET_RX_START	18
PACKET_RX_UPDATE	19
PACKET_RX_STOP	20
HOPPING_DWELL_TIME	21
LE_TX_DUT_TEST_CMD	22
LE_RX_DUT_TEST_CMD	23
LE_DUT_TEST_END_CMD	24
READ_EFUSE_DATA	25
LE_ CONTINUE _TX _START	28
LE CONTINUE _TX _STOP	29
FW_PACKET_TX_START	30
FW_PACKET_TX_STOP	31
FW_PACKET_RX_START	32
FW_PACKET_RX_STOP	33
FW_CONTINUE_TX_START	34
FW_CONTINUE_TX_STOP	35
FW_LE_CONTINUE_TX_START	36

FW_LE_CONTINUE_TX_STOP	37
FW_READ_TX_POWER_INFO	38

Table 13: The parameter Indexes define in "bt_mp_Exec" Table

INDEX	VALUE	Length (Byte)	Value Range	Table Index
0	PGRawData	256	Row data	None
1	ChannelNumber	1	0~78	None
2	PacketType	1	0~9	Table 21: The parameter Indexes define in PacketType Table
3	PayloadType	1	0~7	Table 22: The parameter Indexes define in PayloadType Table
4	TxPacketCount (only for packet tx)	2	0~0xFFF	0:infinite TX packet count
6	WhiteningCoeffValue	1	0x00~0x7F	0x00~0x7F: Enable Whitening 0x80~0xFF: Disable Whitening
7	TxGainIndex	1	1~7	None
9	PacketHeader	4	0x0~0x3FFFF	None
10	HoppingFixChannel (for Hopping mode)	1	0 : Disable 1 : Enable Fix Channel	None
11	HitTarget	6	6 bytes	None
14	Xtal	4	0~0x3F	None
15	LEDataLen	1	0~0x25	None

Table 14: The parameter Indexes define in “bt_mp_setParam” Table

NAME	INDEX	Payload Length in bits
BT_PKT_DH1	0	0~27*8
BT_PKT_DH3	1	0~183*8
BT_PKT_DH5	2	0~339*8
BT_PKT_2DH1	3	0~54*8
BT_PKT_2DH3	4	0~367*8
BT_PKT_2DH5	5	0~679*8
BT_PKT_3DH1	6	0~83*8
BT_PKT_3DH3	7	0~552*8
BT_PKT_3DH5	8	0~1021*8
BT_PKT_LE	9	0~39*8

Table 15: The parameter Indexes define in PacketType Table

NAME	INDEX
BT_PAYLOAD_TYPE_ALL0	0
BT_PAYLOAD_TYPE_ALL1	1
BT_PAYLOAD_TYPE_0101	2
BT_PAYLOAD_TYPE_1010	3
BT_PAYLOAD_TYPE_0x0_0xF	4
BT_PAYLOAD_TYPE_0000_1111	5
BT_PAYLOAD_TYPE_1111_0000	6
BT_PAYLOAD_TYPE_PRBS9	7

Table 16: The parameter Indexes define in PayloadType Table

NAME	INDEX	RETURN					
PKT TX	1	Status	TXBits	TxCounts			
CONT TX	2	Status	TXBits	TxCounts			
PKT RX	3	Status	RxRssi	RXBits	RxCounts	RxErrorBits	
Tx Gain Table	4	Status	Tx Gain Table				
Tx DAC Table	5	Status	Tx DAC Table				
Xtal	6	Status	Xtal				
Thermal	7	Status	Thermal				
Stage	8	Status	Stage				
Efuse	10	Status	Efuse				
LE RX	11	Status	RxCounts				
LE CONT TX	12	Status	TXBits	TxCounts			
FW_PKT_TX	13	Status	TXBits	TxCounts			
FW_CONT_TX	14	Status	TXBits	TxCounts			
FW_PKT_RX	15	Status	RxRssi	RXBits	RxCounts	RxErrorBits	
FW_LE_CONT_TX	16	Status	TXBits	TxCounts			
TX_POWER_INFO	17	Status	Max tx power index	1M default tx power index	2M default tx power index	3M default tx power index	LE default tx power index

Table 17: The parameter Indexes define in "bt_mp_Report" Table

Step 6: Stop Packe Tx

bt_mp_Exec 14

If you need to test other parameters, please stop packet tx and go back to step 3

3.2.2. Verify Bluetooth Rx Performance

Measure the DUT Rx sensitivity to check Rx performance is ok or not. The Rx performance test can be measured in Signaling mode (ex: Anritsu 8852B, Agilent N4010A) or Non-Signaling mode (ex: LitePoint IQFlex). Bluetooth Rx criterion is shown as below:

Verify Rx	Test Item	Packet Type	Criterion
			Bluetooth Spec.
	Sensitivity	DH1 or 3DH5	< -70dBm

Table 18: The recommended test items of Bluetooth Rx

EX:

Step 1: Enter MP Mode and download patch code

```
root@android:/ # rtlbtmp
rtlbtmp

::::::::::::::::::::::::::::::::::::::::::::::::::
:::::::: Bluetooth MP Test Tool Starting ::::::::
> enable uart:/dev/ttyS0
enable uart:/dev/ttyS0
> > > enable[Success:0]
```

Step 2: Set default table

```
> bt_mp_Exec 5
bt_mp_Exec 5
bt_mp_Exec[Success:0]
> bt_mp_Exec,5,0x00


> bt_mp_Exec 6
bt_mp_Exec 6
bt_mp_Exec[Success:0]
> bt_mp_Exec,6,0x00
```

Step 3: Set Parameter

Test Item		adb command
Channel	Packet type	PayloadType=PRBS9; WhiteningCoeffValue = 0xFF(disable); PacketHeader=0x3FFFF; HitTarget= 0x000000c6967e
6	DH1	bt_mp_SetParam 1,0x06;2,0x00;3,0x07;6,0xFF;9,0x3FFFF;11,0x000000c6967e
42	DH1	bt_mp_SetParam 1,0x2a;2,0x00;3,0x07;6,0xFF;9,0x3FFFF;11,0x000000c6967e
70	DH1	bt_mp_SetParam 1,0x46;2,0x00;3,0x07;6,0xFF;9,0x3FFFF;11,0x000000c6967e
6	3DH1	bt_mp_SetParam 1,0x06;2,0x06;3,0x07;6,0xFF;9,0x3FFFF;11,0x000000c6967e
42	3DH1	bt_mp_SetParam 1,0x2a;2,0x06;3,0x07;6,0xFF;9,0x3FFFF;11,0x000000c6967e
70	3DH1	bt_mp_SetParam 1,0x46;2,0x06;3,0x07;6,0xFF;9,0x3FFFF;11,0x000000c6967e

Step 4: To setting Parameter with the Bluetooth test instrument. Bluetooth test instrument begin transmit.

Step 5: Run Packe Rx

bt_map_Exec 18

Step 6: Report Received Result.

bt_mp_Report 3

bt_mp_Report 3" should be executed every 1s.

Step 7: Stop Packe Rx

bt_map_Exec 20

If you need to test other parameters, please stop packet Rx and go back to step 3.

4. Write WiFi Data to Storage

There are three methods provided.

1. Writing All Data to Efuse.
2. Read Linux Efuse file Map Load and Mask Map to Driver Fake.
3. When Efuse was Written, Modify a Part of Data to Efuse.

4.1. Writing All Data to Efuse

Use linux command line to write data to efuse. When DUT boot up, driver load efuse for initial DUT. Write board-dependent information into respective eFuse offset, this information include MAC address, calibrated Tx index(the eFuse location is as the above mentioned), Thermal Meter(the eFuse location is as the above mentioned), and so on.

4.1.1. Initial WiFi Step

Please refer to Section 2.2.1.

4.1.2. Write WiFi Efuse

4.1.2.1. Write All Data to Driver Fake

Driver only saves the contents in shadow memory (Driver Fake). Until you write down all data, you can use command (section 3.1.2.6) to write effuse from driver fake. This can increase efuse lifetime.

Below diagram shows content of Mapfile.

```
0x00 ---> 29 81 03 7c 01 08 28 00 42 07 0d 45 10 00 00 00
0x10 ---> 28 28 27 27 27 29 31 31 31 30 30 e0 ff ff ff ff
0x20 ---> 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 28 28 27 27 27 29
          31 31 31 30 30 e0 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 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 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
          ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
          ff ff ff ff ff ff ff ff 20 20 1b 00 00 00 ff ff
          ff 39 20 11 00 00 00 ff 00 ff 12 ff ff ff ff ff
```

Figure 12: Mapfile of Linux Format

And use below command to write eFuse content to driver fake.

```
iwpriv wlan0 efuse_set wlwfake,0x0,2981037c0108280042070d4510000000
iwpriv wlan0 efuse_set wlwfake,0x10,2828272727293131313030e0ffffffff
iwpriv wlan0 efuse_set wlwfake,0x20,ffffffffffffffffffffffffffffffff
iwpriv wlan0 efuse_set wlwfake,0x30,ffffffffffffffffffffff2828272729
iwpriv wlan0 efuse_set wlwfake,0x40,3131313030e0ffffffffffffffffffff
iwpriv wlan0 efuse_set wlwfake,0x50,ffffffffffffffffffffffffffffffff
iwpriv wlan0 efuse_set wlwfake,0x60,ffffffffffffffffffffffffffffffff
iwpriv wlan0 efuse_set wlwfake,0x70,ffffffffffffffffffffffffffffffff
iwpriv wlan0 efuse_set wlwfake,0x80,ffffffffffffffffffffffffffffffff
iwpriv wlan0 efuse_set wlwfake,0x90,ffffffffffffffffffffffffffffffff
iwpriv wlan0 efuse_set wlwfake,0xa0,ffffffffffffffffffffffffffffffff
iwpriv wlan0 efuse_set wlwfake,0xb0,ffffffffffffffff20201b000000ffff
iwpriv wlan0 efuse_set wlwfake,0xc0,ff392011000000ff00ff12ffffffffff
```

Figure 13: Write Efuse CMD

4.1.2.2. Write Crystal to Driver Fake

Use below command to write one byte data to driver fake.
The relative control commands about modifying WiFi Crystal.

rtwpriv wlan0 efuse_set wlwfake,B9,Crystal_value (Please write hexadecimal value)

ex: Please refer to section 2.3.1.2 you can get Crystal value (the value is decimal).
If Crystal value = 2, you use command as below:

rtwpriv wlan0 efuse_set wlwfake,B9,02

4.1.2.3. Write Thermal to Driver Fake

Use below command to write one byte data to driver fake.
The relative control commands about modifying WiFi Thermal.

rtwpriv wlan0 efuse_set wlwfake,BA,thermal_value (Please write hexadecimal value)

ex: Please refer to section 3.3.2.4, you can get thermal value (the value is decimal).
If thermal value = 29, you use command as below:

rtwpriv wlan0 efuse_set wlwfake,BA,1D

4.1.2.4. Write WiFi Mac Address to Driver Fake

The eFuse content about setting of MAC address is as following table.

- **RTL8723DE MAC address**

WiFi MAC address	0XD0

Table 19: RTL8723DE WiFi MAC address offset in eFuse

- **RTL8723DU MAC address**

WiFi MAC address	0X107

Table 20: RTL8723DU WiFi MAC address offset in eFuse

- **RTL8723DS MAC address**

WiFi MAC address	0X11A

Table 21: RTL8723DS WiFi MAC address offset in eFuse

The relative control commands about modifying WiFi MAC Address.
EX: RTL8723DE

rtwpriv wlan0 efuse_set wlwfake,D0,MACaddress

(ex: MAC = 123456789ABC)

rtwpriv wlan0 efuse_set wlwfake,D0, 123456789ABC

4.1.2.5. Read the Driver Fake Map for Verify and Confirm It

When you complete all data which you want to modify, you can use command as below to verify and confirm it.

rtwpriv wlan0 efuse_get wlrfkmap

4.1.2.6. Write Driver Fake to EFuse

When you complete all data which you want to modify, you can use command as below to write driver fake to efuse.

Use command as show:

rtwpriv wlan0 efuse_set wlfk2map

to write driver fake to eFuse.

4.1.3. Read WiFi EFuse

Use command as show:

rtwpriv wlan0 efuse_get realmap

to read eFuse.

4. 2. Read Linux Efuse file Map Load to Driver Fake (Recommend)

If you want to read efuse file to driver fake, you can use follow command.

4.2.1. Initial WiFi Step

Please refer to Section 2.2.1.

4.2.2. Read Linux Efuse file Map Load to Driver Fake

Open effuse map file and loading to driver fake map

Use command as show:

rtwpriv wlan0 efuse_file /../../efuse.map

```
29 81 00 CC 0B 00 00 00 00 0C 04 4C 10 0C 00 00
20 20 20 20 20 20 26 26 26 26 26 02 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 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 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 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 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 FF FF FF FF FF FF 20 20 1A 00 00 00 00 FF
FF 11 00 10 00 FF 00 FF 00 00 FF FF FF FF FF FF
DA 0B 79 F1 42 66 40 00 E0 4C F1 79 00 09 03 52
65 61 6C 74 65 6B 09 03 38 30 32 2E 31 31 6E 00
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

Figure 14: efuse.map contents

Attention: **If the driver version over the "rtl8xxx_WiFi_linux_ v5.X.X_19292". Pair of the load efuse file and mask_file CMD before the wlfk2map.**

rtwpriv wlan0 efuse_file /…/…/xxx.map

rtwpriv wlan0 efuse_set wlfk2map

If there is no mask file be loaded, it will return error message.

rtwpriv wlan0 efuse_mask /xx/xx/xxmask.txt

Or Use input eFuse mask data:

rtwpriv wlan0 efuse_mask data,00:11:22:33:44:55:66:77:88:99:FF:AA:BB:CC:DD:EE

Return message format:

wlan0 efuse_mask:data:00:11:22:33:44:55:66:77:88:99:ff:aa:bb:cc:dd:ee

Load Efuse Mask data 16 hex ok

4.2.2.1. Read the Driver Fake Map for Verify and Confirm It

Use command as show:

rtwpriv wlan0 efuse_get wlrfkmap

4.2.2.2. Modify a Part of Data to Driver Fake

Use command as show:

rtwpriv wlan0 efuse_set wlwfake,C9,012345…

Address C9 and value 01234 is an example.

4.2.2.3. Write Driver Fake to EFuse

When you complete data which you want to modify, you can use command as below to write driver fake to efuse.

Use command as show:

rtwpriv wlan0 efuse_set wlfk2map

to write driver fake to eFuse.

4.2.2.4. Read WiFi EFuse

Use command as show:

rtwpriv wlan0 efuse_get realmap

to read eFuse.

4.3. When Efuse Was Written, Modify a Part of Data to Efuse

When Efuse was written, you want to modify a part of data to eFuse. You can use below step.

Note: If efuse is empty, don't use this method.

4.3.1. Initial WiFi Step

Please refer to Section 2.2.1.

4.3.2. Writing a Part of Data to Efuse

4.3.2.1. Read Efuse Map and Loading to Driver Fake Map

Use command as show:

rtwpriv wlan0 efuse_set wldumpfake

to read efuse map and loading to driver fake map

4.3.2.2. Modify a Part of Data to Driver Fake

Use command as show:

rtwpriv wlan0 efuse_set wlwfake,C9,012345…

Address C9 and value 01234 is an example.

4.3.2.3. Read the Driver Fake Map for Verify and Confirm It

When you complete all data which you want to modify, you can use command as below to verify and confirm it.

rtwpriv wlan0 efuse_get wlrfkmap

4.3.2.4. Write Driver Fake to EFuse

When you complete data which you want to modify, you can use command as below to write driver fake to efuse.

Use command as show:

rtwpriv wlan0 efuse_set wlfk2map

to write driver fake to eFuse.

4.3.2.5. Read WiFi EFuse

Use command as show:

rtwpriv wlan0 efuse_get realmap

to read eFuse.

4.3.2.6. Another Method Writes Data to Efuse

When efuse is empty or efuse has data, you can use command as show:

rtwpriv wlan0 efuse_set wmap,C9,0123456789…

This method writes efuse immediate, not use driver fake.

Address C9 and value 0123456789… is an example.

5. Write Bluetooth Data to Storage

There are two methods provided.

1. Writing All Data to Efuse.
2. Read Linux Efuse file Map Load and Mask Map to Driver Fake.

5.1. Writing All Data to Efuse

Use linux command line to write data to efuse. When DUT boot up, driver load efuse for initial DUT. Write board-dependent information into respective eFuse offset, this information include MAC address, calibrated Tx index(the eFuse location is as the above mentioned), and so on.

5.1.1. Write Bluetooth Efuse

5.1.1.1. Write All Data to Driver Fake

Driver only saves the contents in shadow memory (Driver Fake). Until you write down all data, you can use command to write effuse from driver fake. This can increase efuse lifetime.

rtwpriv wlan0 efuse_set btwfake,14,0123456789…
…
…

5.1.1.2. Write Driver Fake to EFuse

When you complete all data which you want to modify, you can use command as below to write driver fake to efuse.

Use command as show:

echo 1 > /sys/class/rfkill/rfkill0/state //bt power on
rtwpriv wlan0 effuse_set btfk2map

to write driver fake to eFuse.

5.1.1.3. Read the Driver Fake Map for Verify and Confirm It

When you complete all data which you want to modify, you can use command as below to verify and confirm it.

rtwpriv wlan0 efuse_get btfmap //Read from HW BT of the front efuse logic map.
rtwpriv wlan0 efuse_get btbmap //Read from HW BT of the back efuse logic map.

Read the contents of the fake before and after.

5.2. When Efuse Was Written, Modify a Part of Data to Efuse

When Efuse was written, you want to modify a part of data to eFuse. You can use below step.
Note: If efuse is empty, don't use this method.

5.2.1. Writing a Part of Data to Efuse

5.2.1.1. Read Efuse Map and Loading to Driver Fake Map

Use command as show:

rtwpriv wlan0 efuse_set btdumpfake

to read efuse map and loading to driver fake map

5.2.1.2. Modify a Part of Data to Driver Fake

Use command as show:

rtwpriv wlan0 efuse_set btwfake,14,0123456789…

5.2.1.3. Write Driver Fake to EFuse

When you complete all data which you want to modify, you can use command as below to write driver fake to efuse.

Use command as show:

echo 1 > /sys/class/rfkill/rfkill0/state //bt power on
rtwpriv wlan0 effuse_set btfk2map

to write driver fake to eFuse.

5.2.1.4. Read the Driver Fake Map for Verify and Confirm It

When you complete all data which you want to modify, you can use command as below to verify and confirm it.

rtwpriv wlan0 efuse_get btfmap //Read from HW BT of the front efuse logic map.

rtwpriv wlan0 efuse_get btbmap //Read from HW BT of the back efuse logic map.

Read the contents of the fake before and after.

6. K-FREE FLOW

When you finish Chap1~Chap4, you will get sufficient mapfile. These mapfile include calibrated Crystal ,Tx power index ,Thermal meter offset and Power Difference, and these value must be averaged for created new mapfile. You write new mapfile to eFuse, and restart DUT to apply default value. And then verify Tx/Rx performance.

6.1. Prepared Job

When you finish Chap1~Chap4, you will get sufficient mapfile. Please refer to Tabel1 and Table2 to the location of Crystal ,Tx power index ,Thermal meter offset and Power Difference in mapfile. You must average these value and update mapfile. And then write the map file to efuse.

6.2. Write MAP to EFuse

6.2.1. Write Map to Storage and Check Contents

Below diagram shows the first step of kfree flow.

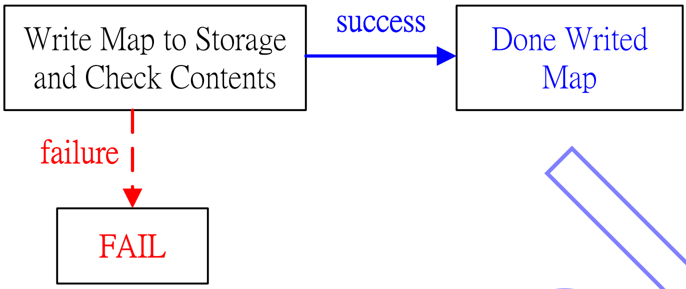

Figure 15: Write MAP to EFuse Flow

6.2.2. Initial DUT

Please refer to Section 2.2.1.

The command about initial DUT is as below

remount
root
rmmod wlan
insmod wlan.ko
(delay 5 second)
ifconfig wlan0 up
rtwpriv wlan0 mp_start

6.2.3. Write MAP to EFuse

Please refer to Section 3.2. The command about writing eFuse is as below.

Step1:
Read map file and mask data.

rtwpriv wlan0 efuse_file /…/…/xxx.map
rtwpriv wlan0 efuse_mask /xx/xx/xxmask.txt

Step 2:
Read the Driver Fake Map for Verify and Confirm It

rtwpriv wlan0 efuse_set wlfk2map

Step 3:
Modify a Part of Data to Driver Fake
EX: Mac Address,

The eFuse content about setting of MAC address is as following table.

- **RTL8723DE MAC address**

WiFi MAC address	0XD0

Table 22: RTL8723DE WiFi MAC address offset in eFuse

- **RTL8723DU MAC address**

WiFi MAC address	0X107

Table 23: RTL8723DU WiFi MAC address offset in eFuse

- **RTL8723DS MAC address**

WiFi MAC address	0X11A

Table 24: RTL8723DS WiFi MAC address offset in eFuse

The relative control commands is about modifying WiFi MAC Address.
EX: RTL8723DE

rtwpriv wlan0 efuse_set wlwfake,D0,MACaddress

Step 4:
Read the Driver Fake Map for Verify and Confirm It
When you finish the modification, you can use command as below to verify and confirm it.

rtwpriv wlan0 efuse_get wlrfkmap

Step 5:
Write Driver Fake to EFuse
When you finish the modification, you can use command as below to write driver fake to efuse.

Use command as show:

rtwpriv wlan0 efuse_set wlfk2map

to write driver fake to eFuse.

Step 6:
Read WiFi EFuse
Use command as show:

rtwpriv wlan0 efuse_get realmap

to read eFuse.

6.3. Verify Tx/Rx Performance

Below diagram shows second step of kfree flow.

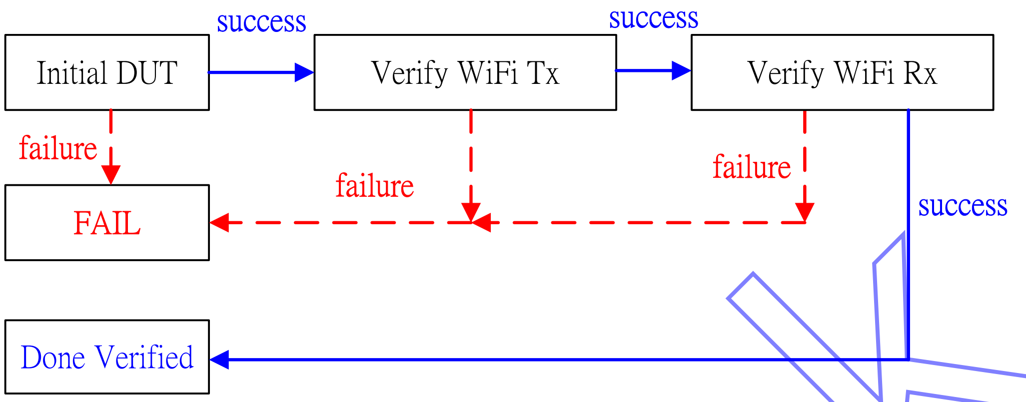

Figure 16: Verify Tx/Rx Performance Flow

6.3.1. Initial DUT

If we write Map to eFuse successfully. We must initial DUT for verified.

The relative control commands about Initial DUT.

remount
root
rmmod wlan
insmod wlan.ko
(delay 5 second)
ifconfig wlan0 up
rtwpriv wlan0 mp_start

6.3.2. Verify Tx/Rx Performance

We must verify WiFi Tx/Rx Performance .Please refer to Chap3, Chap4.

The command of verifying WiFi Tx Performance is as below:
Verify WiFI Tx Performance
Ex: Rate **MCS7**, Bandwidth**40**, Channel **4** , Tx_target = 13dBm, **Index_MCS7_B40_Group2 = 42(the value is from map)**

rtwpriv wlan0 mp_bandwidth 40M=1,shortGI=0
rtwpriv wlan0 mp_channel 4

rtwpriv wlan0 mp_rate 135
rtwpriv wlan0 mp_txpower patha=42,pathb=0
rtwpriv wlan0 mp_ctx background,pkt

Verify WiFI Rx Performance

ex: Start Rx Test(use 802.11b, channel 1, **11**Mbps, Antenna 0, Bandwidth 20)

rtwpriv wlan0 mp_bandwidth 40M=0,shortGI=0
rtwpriv wlan0 mp_channel 1
rtwpriv wlan0 mp_arx start
rtwpriv wlan0 mp_reset_stats

Stop Rx Test and get report

rtwpriv wlan0 mp_arx mac
rtwpriv wlan0 mp_arx stop