

Realtek

# Bluetooth MP Flow

(windows)

Date: 2017/11/17

**This document is subject to change without notice. The document contains Realtek confidential information and must not be disclosed to any third party without appropriate NDA.**

## 1. Overview

This document is used to introduce MP (Mass Production) test tool for Realtek Bluetooth chip series. Customers should comply with the steps and requirements under this document. Contact Realtek Bluetooth FAE if any problem arises in the use of MP flow.

Realtek

## 2. Prerequisites

Before MP tool startup, below items should be checked:

- a) The Bluetooth Chip embedded is provided by Realtek;

Realtek Bluetooth Chip Number
RTL8723B series
RTL8761A series
RTL8821A series
RTL8703B series
RTL8723D series
RTL8822B series
RTL8821C series

- b) The Bluetooth on target production is available and works normally;

### 2.1 Files

MP tool package is provided to customers in binary format:

RtlBluetoothMP.dll	MP library
RtlBluetoothMP.h	MP header
BTPatchCode\ Patch_rtl8723a.bin	rtl8723a fw patch
BTPatchCode\ Patch_rtl8723b.bin	rtl8723b fw patch
BTPatchCode\ Patch_rtl8821a.bin	rtl8821a fw patch
BTPatchCode\ Patch_rtl8761a.bin	rtl8761a fw patch
BTPatchCode\ Patch_rtl8703b.bin	rtl8703b fw patch
BTPatchCode\ Patch_rtl8723d.bin	rtl8723d fw patch
BTPatchCode\ Patch_rtl8822b.bin	rtl8822b fw patch
BTPatchCode\ Patch_rtl8821c.bin	rtl8821c fw patch
BTPatchCode\ Patch_rtl8822c.bin	rtl8822c fw patch
BTPatchCode\ Patch_rtl8761b.bin	rtl8761b fw patch

## 3. MP API's function Usage

### 3.1 Use the customer's interface: BTMPAPI\_BuildInterfaceVendor

```
int BTMPAPI_BuildInterfaceVendor(  
    BASE_INTERFACE_MODULE **ppBaseInterface,  
    BASE_INTERFACE_MODULE *pBaseInterfaceModuleMemory,  
    //Parmater  
    unsigned int InterfaceType,  
    unsigned char PortNo,  
    unsigned long Baudrate,  
    //basic fuction  
    BASE_FP_OPEN Open,  
    BASE_FP_SEND Send,  
    BASE_FP_RECV Recv,  
    BASE_FP_CLOSE Close,  
    BASE_FP_WAIT_MS WaitMs  
);
```

This function provides vender to implement how to open interface, close interface, send hci command, receive hci event, and wait mini-second functions by themselves.

### 3.2 Use the RTK's interface: BTMPAPI\_BuildInterfaceRTK

```
int BTMPAPI_BuildInterfaceRTK(  
    BASE_INTERFACE_MODULE **ppBaseInterface,  
    BASE_INTERFACE_MODULE *pBaseInterfaceModuleMemory,  
    unsigned int InterfaceType,  
    unsigned char PortNo,  
    unsigned long Baudrate  
);
```

This function uses Realtek default functions to open, close, send, receive and wait. Therefore, you could select one between BuildInterfaceVendor and BuildInterfaceRTK.

### 3.3 Use the SW API's of RTK: BTMPAPI\_BuildBluetoothModule

The Bluetooth Module includes four main functions that are “DownloadPatchCode”, “UpDataParameter”, “ActionControlExcute”, and “ActionReport”.

### 3.3.1 DownloadPatchCode

```
typedef int
(*BT_DLL_MODULE_FP_ACTION_DLFW)(
    BASE_BTMPDLL_MODULE *pDLLBtBaseModule,
    char *pPatchcode,
    int patchLength,
    int Mode
);
```

This function is used to download fw patch. Before downloading fw patch, vendor should read different patch files according to different BT chips.

### 3.3.2 UpDataParameter

```
typedef int
(*BT_DLL_MODULE_FP_UPDATA_PARAMETER)(
    BASE_BTMPDLL_MODULE *pDLLBtBaseModule,
    BT_PARAMETER *pParam
);
```

This function is used to pass “BT\_PARAMETER” data structure to RtlBluetoothMP.dll. However, BT hardware registers don’t be changed until “ActionControlExcute” is called. The following tables show more detail information about each parameter.

Name	Value Range	Table Index
ParameterIndex	See <b>BT_ACTION_CONTROL</b>	See <b>BT_ACTION_CONTROL</b>
mPGRawData	Row data	None
mChannelNumber	0~78	None
mPacketType	0~8	See <b>PKT_TYPE</b>
mTxGainIndex	1~MAX_POWER_INDEX	Depend chip ,please see table <b>MAX_TXPOWER_INDEX</b>
mTxGainValue	0xFF(Realtek define)	Realtek define
mTxPacketCount (for packet tx)	0~0xFFF	0 : infinite Tx packet count
mPayloadType	0~7	See <b>PAYLOAD_TYPE</b>
mPacketHeader	0x0~0x3FFFF	See <b>PACKET_HEADER</b>
mWhiteningCoeffValue	0x00~0x7F	0x00~0x7F : Enable Whitening 0x80: Disable Whitening
mTxDAC	Realtek define	Realtek define
mHitTarget	6 bytes	None
bHoppingFixChannel (for Hopping mode)	0 : Disable 1 : Enable Fix Channel	None

Rtl8761Xtal	0~0x7F	Depend Chip
LEDataLen	1	0~0x255 (BT4.0 : 0x00~0x0x25)
PHY	1~4	See BLE_PKT_TYPE
ModulationIndex	0,1	See BLE_MODULATION_TYPE

**Table BT\_PARAM**

The packet types are defined in Table PKT\_TYPE:

Legacy:

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
BT_PKT_TYPE_NULL	10	None
BT_PKT_LE_2M	11	0~255
BT_PKT_LE_CODED_S8	12	0~255
BT_PKT_LE_CODED_S2	13	0~255

**Table PKT\_TYPE**

BLE 5.0 Tx :NAME	INDEX
BLE5_TX_1M_PHY	1
BLE5_TX_2M_PHY	2
LE5_TX_CODED_PHY_S8	3
LE5_TX_CODED_PHY_S2	4
BLE 5.0 RX NAME	INDEX
BLE5_TX_1M_PHY	1
BLE5_TX_2M_PHY	2
LE5_TX_CODED_PHY_LR	3

**Table BLE\_PKT\_TYPE**

NAME	INDEX	
STANDARD_MODULATION_INDEX	0	
STABLE_MODULATION_INDEX	1	

**Table BLE\_MODULATION\_TYPE**

The payload types are defined in Table PAYLOAD\_TYPE.

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 PAYLOAD\_TYPE**

Packet type	Payload(Bits)	Packet Header Hex
<b>DH1</b>	216	<b>33820</b>
<b>DH3</b>	1464	<b>39858</b>
<b>DH5</b>	2712	<b>A078</b>
<b>2DH1</b>	432	<b>33820</b>
<b>2DH3</b>	2936	<b>C050</b>
<b>2DH5</b>	5432	<b>3F870</b>
<b>3DH1</b>	664	<b>15C40</b>
<b>3DH3</b>	4416	<b>39858</b>
<b>3DH5</b>	8168	<b>A078</b>

**Table PACKET\_HEADER**

For Execute Command defines:

Command	Index	Support Chip									
		RTL8723B	RTL8761A	RTL8821A	RTL8723D	RTL8822B	RTL8821C	RTL8763B	Ameba D/ZII	RTL8822C	RTL8761B
HCI_RESET	0	■	■	■	■	■	■	■	■	■	■
TEST_MODE_ENABLE	1	■	■	■	■	■	■	■	■	■	■
WRITE_EFUSE_DATA	2	■ (RTL8761A)									■
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		■	■	■	■	■	■	■	■
SET_GPIO3_0	39	■ (RTL8761A)	■							
SET_ANT_INFO	40		■					■		
SET_ANT_DIFF_S0S1	41		■							
TX_POWER_TRACKING	42		■	■	■	■	■	■	■	■
SET_K_TX_CH_PWR	43		■	■				■		
WRITE_FLASH_CONFIG	44					■				
TX_POWER_GAIN_K	45					■	■	■	■	■
TX_POWER_FLATNESS	46					■	■	■	■	■



TX_PATH_LOSS_MODULE	47					■	■	■	■	■
CONFIG_EXTEND(Flash write)	48					■	■			
UNLOCK_8822C	49						■			
SetTxPower_8822C	50					■	■	■	■	■

**Table BT\_ACTIONCONTROL\_TAG**

- *GET\_MAX\_TXPOWER\_INDEX*

GET\_MAX\_TXPOWER\_INDEX is a special Vendor command. This command is able to get device maximum Tx power index(eg. *MAX\_TXPOWER\_INDEX*).

- *TxPacketCount parameter*

TxPacketCount is used to set how many TX packets will be transmitted. The range of TxPacketCount is from 0 to 0x3FFFF. However, if TxPacketCount equals to “0”, it means infinite TxPacketCounts. If users want to keep sending packet tx, “TxPacketCount” should be “0” and “bt\_mp\_Report 1” should be executed every 200~1000ms. To execute “bt\_mp\_Report 1” periodically can trigger bt hardware to send packets.

- *WhiteningCoeffValue parameter*

The range of WhiteningCoeffValue is from 0 to 0x7F. However, if WhiteningCoeffValue is “0x80”, it means to disable whitening.

- *TxGainIndex and TxGainValue parameter*

TxGainIndex and TxGainValue are both to set TX power gain. However, TxGainIndex have higher priority. The range of TxGainIndex is from 1 to *MAX\_TXPOWER\_INDEX*. Only when TxGainIndex is out of range, TxGainValue is meaningful.

Realtek Bluetooth Chip Number	<i>MAX_TXPOWER_INDEX</i>
RTL8723B series	7
RTL8761A series	7
RTL8821A series	7
RTL8703B series	7
RTL8723D series	<i>GET_MAX_TXPOWER_INDEX Command</i>
RTL8822B series	<i>GET_MAX_TXPOWER_INDEX Command</i>
RTL8821C series	<i>GET_MAX_TXPOWER_INDEX Command</i>
Other	<i>GET_MAX_TXPOWER_INDEX Command</i>

**Table MAX\_TXPOWER\_INDEX**

### 3.3.3 ActionControlExcute

```
typedef int
(*BT_DLL_MODULE_FP_ACTION_CONTROLEXCUTE)(
    BASE_BTMPDLL_MODULE *pDLLBtBaseModule
);
```

According to “ParameterIndex” in the “BT\_PARAMETER” structure, “ActionControlExcute” performs different functions, such as “PACKET\_TX\_START”, “PACKET\_TX\_STOP”, etc.

### 3.3.4 ActionReport

```
typedef int
(*BT_DLL_MODULE_FP_ACTION_REPORT)(
    BASE_BTMPDLL_MODULE *pDLLBtBaseModule,
    int ActiceItem,
    BT_DEVICE_REPORT *pReport
);
```

This function is used to report current TX/RX packet counts and chip status. The following table shows the relation between “ActionItem” and the parameters in “BT\_DEVICE\_REPORT”. For example, TotalTXBits and TotalTxCounts values are updated when “ActionItem” equals to “REPORT\_PKT\_TX”.

ActiceItem	Index	BT_DEVICE_REPORT
REPORT_PKT_TX	1	TotalTXBits TotalTxCounts
REPORT_CON_TX	2	TotalTXBits TotalTxCounts
REPORT_RKT_RX	3	RXRecvPktCnts TotalRXBits TotalRxCounts TotalRxErrorBits RxRssi ber Cfo
REPORT_TX_GAIN_TABLE	4	CurrTXGainTable
REPORT_TX_DAC_TABLE	5	CurrTXDACTable
REPORT_XTAL	6	CurrRtl8761Xtal
REPORT_CHIP	9	pBTInfo BTInfoMemory
REPORT_LOGICAL_EFUSE	10	ReportData
REPORT_LE_RX	11	TotalRxCounts

### 3.4 Use the SW API's to enter BT HCI Mode:BTMPAPI\_ExternFunctionRTK

If you need to send a special string to enter the BT interface then can use the API below to ENTER.

RTKBTMP\_API

```
int BTMPAPI_ExternFunctionRTK(  
    int Chip_type,  
    int Enter1_Leave2,  
    BASE_INTERFACE_MODULE **ppBaseInterface,  
    BASE_INTERFACE_MODULE *pBaseInterfaceModuleMemory,  
    unsigned int    InterfaceType,  
    unsigned char   PortNo,  
    unsigned long   Baudrate,  
    unsigned char *pWriting,  
    unsigned char *pReading,  
    unsigned int    return_check,  
    unsigned char   *pData,  
    unsigned int    Ending_close  
);  
Enter string
```

Chip_type	Item	String
RTK_BT_CHIP_ID_AMEBAD = 15	Enter	"ATM0=bridge,baudrate0,baudrate1,parity\n\r"
RTK_BT_CHIP_ID_AMEBAZII =16		

Leave string:

Chip_type		String
RTK_BT_CHIP_ID_AMEBAD = 15	Leave	"ATM0=bridge,baudrate0,baudrate1,parity\n\r"
RTK_BT_CHIP_ID_AMEBAZII =16		

For Ameba D/ZII Sample, you can refer below:

```
// Enter BT HCI Mode
```

```
Int rtn=ExternFunctionRTK(, RTK_BT_CHIP_ID_AMEBAD
```

```
1, //ener = 1,leave =2
```

```
NULL, NULL,
```

```
InterfaceType,
```

```
PortNo,
```

```
Baudrate,
```

```
NULL, NULL,
```

```
1, //need check event
```

```
NULL, // extern data
```

```
1); // send after close port
```

## 4. RF Test Mode (Certification)

### 4.1 BR/EDR–Direct Test Mode (Certification)

Enter the Bluetooth device to Bluetooth DUT Test and stop DUT Test Mode command below:

--Enter DUT Test Mode:

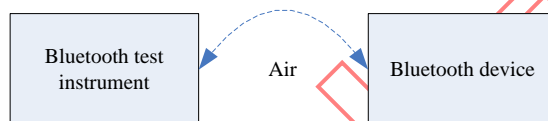
```
pParam->ParameterIndex = TEST_MODE_ENABLE;  
pBluetoothModule->UpDataParameter(pBluetoothModule,pParam);  
pBluetoothModule->ActionControlExcute(pBluetoothModule);
```

--Stop Test Mode

```
pParam->ParameterIndex = HCI_RESET;  
pBluetoothModule->UpDataParameter(pBluetoothModule,pParam);  
pBluetoothModule->ActionControlExcute(pBluetoothModule);
```

Use bt\_mp\_Exec(HCI\_RESET) to stop Bluetooth test mode.

The Test connection diagram :



### 4.2 TX Test(Certification)

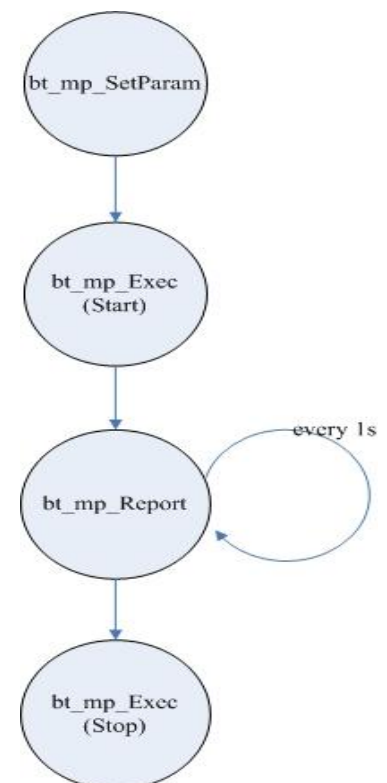
In order to transmit a continuous signal, follow the process below:

Step1: UpDataParameter

Name
ParameterIndex
mChannelNumber
mPacketType
mPayloadType
mWhiteningCoeffValue
mTxGainIndex
mPacketHeader
mHitTarget

Step2: ActionControlExcute (CONTINUE\_TX\_START)

```
pParam->ParameterIndex = CONTINUE_TX_START;
```



```
pBluetoothModule->UpDataParameter(pBluetoothModule,pParam);
```

```
pBluetoothModule->ActionControlExcute(pBluetoothModule);
```

Step3: ActionReport (REPORT\_CON\_TX)

```
pBluetoothModule->ActionReport(pBluetoothModule, REPORT_CON_TX, pReport);
```

Step4: ActionControlExcute (CONTINUE\_TX\_STOP)

```
pParam->ParameterIndex = CONTINUE_TX_STOP;
```

```
pBluetoothModule->UpDataParameter(pBluetoothModule,pParam);
```

```
pBluetoothModule->ActionControlExcute(pBluetoothModule);
```

PS. "ActionReport" should be called every 200ms.

## 4.3 BLE Direct Test Mode (BQB Test)

After testing BT4.0 only enable device and download patch code, you should jump interface to the instrument.

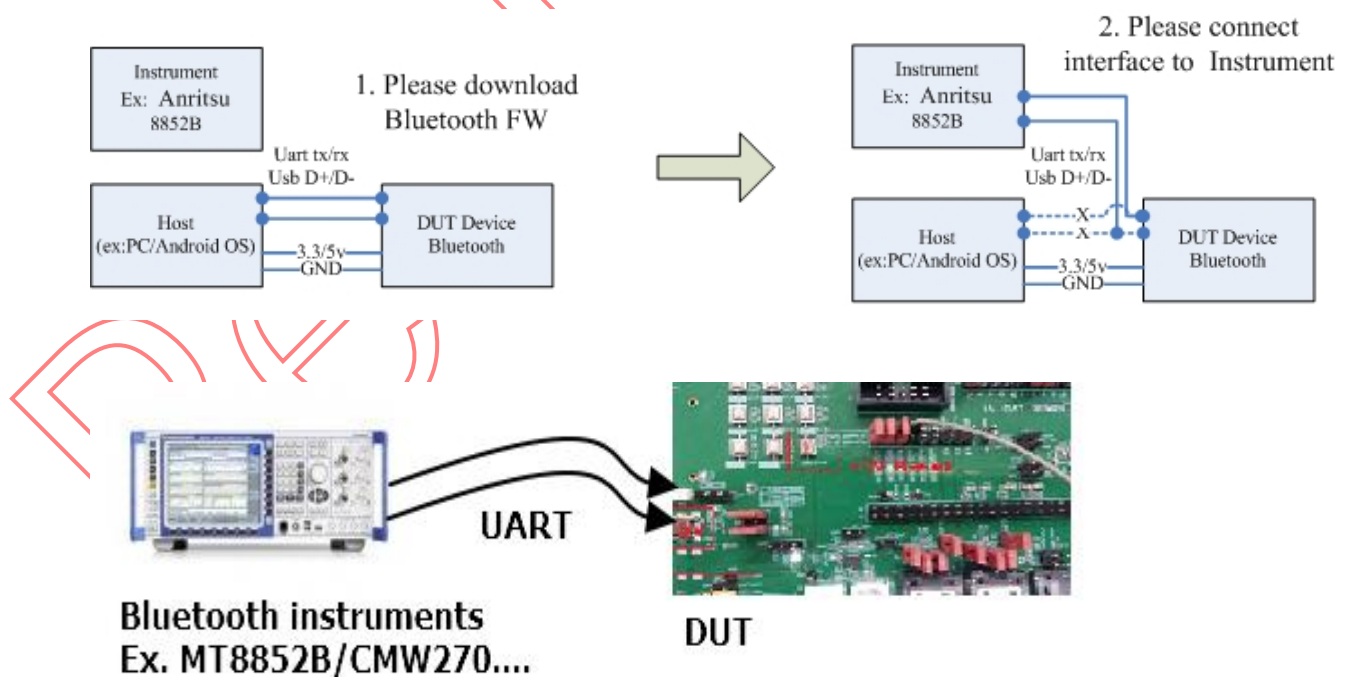
Step 1:Open Device

Step 2:download Bluetooth Patch Code

Step 3: **Jump hardware interface(UART) to the instrument.**

Step 4:Begin test.

BLE is a schematic diagram of the test for Certification:



## 4.4 LE Certification TX Mode(Certification)

In order to facilitate RF safety testing, such as the band edge. We often use to facilitate continuous tx RF safety testing. eg. **Bandedge**

Step1: UpDataParameter ; Channel Number = 0~39

INDEX	VALUE
1	ChannelNumber

Step2: ActionControlExcute (LE\_TX\_START\_CMD)

Step4: ActionControlExcute (LE\_TX\_STOP\_CMD)

## 4.5 Hopping Test (Certification- Dwell time)

In order to start hopping mode test, follow the steps below:

Step 1: bt\_mp\_SetParam....(to setting packet type:DH1,DH2...3DH5)

Name
PacketType
HoppingFixChannel
ChannelNumber
WhiteningCoeffValue

Packet Type	INDEX
BT_PKT_DH1	0
BT_PKT_DH3	1
BT_PKT_DH5	2
BT_PKT_2DH1	3
BT_PKT_2DH3	4
BT_PKT_2DH5	5
BT_PKT_3DH1	6
BT_PKT_3DH3	7
BT_PKT_3DH5	8
BT_PKT_LE	9
BT_PKT_NULL	10

Step 2: bt\_mp\_Exec(HOPPING\_DWELL\_TIME)

```
pParam->ParameterIndex = HOPPING_DWELL_TIME;
```

```
pBluetoothModule->UpDataParameter(pBluetoothModule,pParam);
```

```
pBluetoothModule->ActionControlExcute(pBluetoothModule)
```

Step 3: bt\_mp\_Exec(HCI\_RESET) to disable hopping mode.

```
pParam->ParameterIndex = HCI_RESET;
```

```
pBluetoothModule->UpDataParameter(pBluetoothModule,pParam);
```

```
pBluetoothModule->ActionControlExcute(pBluetoothModule) ;
```

PS. If HoppingFixChannel = 1, it enable fix channel that is controlled by “ChannelNumber”.

If HoppingFixChannel = 0, “ChannelNumber” is useless.

Realtek

## 5. MP Mode Test Control Steps

### 5.1 DUT TX Mode(MP)

In order to transmit a packet signal, follow the process below:

Step1: UpDataParameter

Name
ParameterIndex
mChannelNumber
mPacketType
mPayloadType
mTxPacketCount
mWhiteningCoeffValue
mTxGainIndex
mPacketHeader
mHitTarget

Step2: ActionControlExcute (PACKET\_TX\_START)

pParam->ParameterIndex = PACKET\_TX\_START;

pBluetoothModule->UpDataParameter(pBluetoothModule,pParam);

pBluetoothModule->ActionControlExcute(pBluetoothModule);

Step3: ActionReport (REPORT\_PKT\_TX)

pBluetoothModule->ActionReport(pBluetoothModule, REPORT\_PKT\_TX, pReport);

Step4: ActionControlExcute (PACKET\_TX\_STOP)

pParam->ParameterIndex = PACKET\_TX\_STOP;

pBluetoothModule->UpDataParameter(pBluetoothModule,pParam);

pBluetoothModule->ActionControlExcute(pBluetoothModule);

PS. "ActionReport" should be called every 200ms.

### 5.2 DUT RX Mode(MP)

In order to receive a packet signal, follow the process below:

Step1: UpDataParameter

Name
ParameterIndex
mChannelNumber
mPacketType
mPayloadType
mWhiteningCoeffValue
mPacketHeader



mHitTarget

Step2: ActionControlExcute (PACKET\_RX\_START)

```
pParam->ParameterIndex = PACKET_RX_START;  
pBluetoothModule->UpDataParameter(pBluetoothModule,pParam);  
pBluetoothModule->ActionControlExcute(pBluetoothModule);
```

Step3: ActionReport (REPORT\_PKT\_RX)

```
pBluetoothModule->ActionReport(pBluetoothModule, REPORT_PKT_RX, pReport);
```

Step4: ActionControlExcute (PACKET\_RX\_STOP)

```
pParam->ParameterIndex = PACKET_RX_STOP;  
pBluetoothModule->UpDataParameter(pBluetoothModule,pParam);  
pBluetoothModule->ActionControlExcute(pBluetoothModule);
```

PS. "ActionReport " should be called every 200ms.

### 5.3 LE DUT Test Mode-TX/RX (MP)

To start LE TX DUT test mode, follow the steps below:

Step 1: bt\_mp\_SetParam

INDEX	VALUE	Value Range
1	ChannelNumber	0~39
3	PayloadType	BT_LE_PAYLOAD_TYPE_PRBS9 = 0, BT_LE_PAYLOAD_TYPE_1111_0000 = 1, BT_LE_PAYLOAD_TYPE_1010 = 2, BT_LE_PAYLOAD_TYPE_PRBS15 = 3, BT_LE_PAYLOAD_TYPE_ALL1 = 4, BT_LE_PAYLOAD_TYPE_ALL0 = 5, BT_LE_PAYLOAD_TYPE_0000_1111 = 6, BT_LE_PAYLOAD_TYPE_0101 = 7,
15	LEDataLen	0x00~0x25

Step 2: bt\_mp\_Exec(LE\_TX\_DUT\_TEST\_CMD)

```
pParam->ParameterIndex = LE_TX_DUT_TEST_CMD;  
pBluetoothModule->UpDataParameter(pBluetoothModule,pParam);  
pBluetoothModule->ActionControlExcute(pBluetoothModule);
```

Step 3: bt\_mp\_Exec(LE\_DUT\_TEST\_END\_CMD) to stop LE TX DUT mode.

```
pParam->ParameterIndex = LE_DUT_TEST_END_CMD;  
pBluetoothModule->UpDataParameter(pBluetoothModule,pParam);  
pBluetoothModule->ActionControlExcute(pBluetoothModule);
```

To start LE RX DUT test mode, follow the steps below:

Step 1: bt\_mp\_SetParam

INDEX	VALUE	Value Range
1	ChannelNumber	0~39

Step 2: bt\_mp\_Exec(LE\_RX\_DUT\_TEST\_CMD)

```
pParam->ParameterIndex = LE_RX_DUT_TEST_CMD;  
pBluetoothModule->UpDataParameter(pBluetoothModule,pParam);  
pBluetoothModule->ActionControlExcute(pBluetoothModule);
```

**Step 3: bt\_mp\_Exec(LE\_DUT\_TEST\_END\_CMD) to stop LE RX DUT mode.**

```
pParam->ParameterIndex = LE_DUT_TEST_END_CMD;  
pBluetoothModule->UpDataParameter(pBluetoothModule,pParam);  
pBluetoothModule->ActionControlExcute(pBluetoothModule);
```

**Step 4: bt\_mp\_Report 11**

```
//Update mBT_DEVICE_REPORT.TotalRxCounts  
pBluetoothModule->ActionReport(pBluetoothModule,REPORT_LE_RX,  
&mBT_DEVICE_REPORT);
```

## 6. MP Test Flow

7. Support function by chap is defined as:

Function	RTL8723B	RTL8703B	RTL8821A	RTL8761A	RTL8822B	RT8821C	RTL8822C	RTL8761B	Ameba	BBPRO	BBPRO2	BBLite	Bee	Bee2	Other
Thermal power tracking Enable/Disable	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
Tx gain K							■	■	■	■	■	■	■	■	■
Tx flatness K							■	■	■	■	■	■	■	■	■
Tx Path loss							■	■	■	■	■	■	■	■	■
Set Tx Power level							■	■	■	■	■	■	■	■	■

### 6.1 Mass Production Test Flow by Non-signaling Mode

In the initial stage, the Bluetooth device must set to factory default.

#### 6.1.1 Disable TX power tracking

If platform is Windows, please use BT DLL of Realtek to set it.

A simple example to use “API\_ThermalPowerTracking” of RF dynamic link library is as follows:

```

pParam->ParameterIndex = TX_POWER_TRACKING;
pParam->mParamData[0] = 0; //set
pParam->mParamData[1] = OnOff; //on(1) or off(0)
pBluetoothModule->UpDataParameter(pBluetoothModule,pParam);
if (pBluetoothModule->ActionControlExcute(pBluetoothModule) !=0){
    goto error;
}

```

You can refer to “BTDevice\_SetThermalPowerTracking(unsigned char OnOff) “ function of sample program.

#### 6.1.2 Set TX gain cal (K) value

If you need to calibrate to targeted power, then must reset the TX gain Cal Value. If platform is Windows, please set it through BT DLL of Realtek.

**The TXGainK Value is a 2's complement.**

Parameter Index	Param Data length	mParamData	
		0	
TX_POWER_GAIN_K	2	0x01 (Set)	TXgain cal Value

A simple example to use “API\_ThermalPowerTracking” of RF dynamic link library is as follows:

```
pParam->ParameterIndex = TX_POWER_GAIN_K;
pParam->mParamData[0] = 0x01; //set
pParam->mParamData[1] = TXgain cal Value; //set value
pBluetoothModule->UpDataParameter(pBluetoothModule,pParam);
if (pBluetoothModule->ActionControlExcute(pBluetoothModule) !=0){
    goto error;
}
```

For Linux/Android platform: In bt rf(MP) test mode, use the following command to complete.

```
bt_mp_SetParam 18,value // The value is Tx Gain k value
bt_mp_Exec 45
```

### 6.1.3 Set TX flatness Value

If you need to calibrate TX power for each frequency, then must set flatness value to 0 first. If platform is Windows, please set it through BT DLL of Realtek.

**The TX flatness Value is a 2's complement.**

Parameter Index	Param Data length	mParamData			
		0	1	2	
TX_POWER_GAIN_K	3	0x02 (Set)	Flatness Value LSB	Flatness Value MSB	

The flatness settings of APIs are defined below:

A simple example to use “API\_ThermalPowerTracking” of RF dynamic link library is as follows:

```
pParam->ParameterIndex = TX_POWER_FLATNESS;
pParam->mParamData[0] = 0x02; //set
pParam->mParamData[1] = TXflatness &0xFF; //LSB;
pParam->mParamData[2] = (TXflatness >>8) &0xFF; //MSB;
pBluetoothModule->UpDataParameter(pBluetoothModule,pParam);
if (pBluetoothModule->ActionControlExcute(pBluetoothModule) !=0){
    goto error;
}
```

### 6.1.4 Set TX path loss module Value

If you need to calibrate TX power to target tx power , then must set the TX path loss module value.

**The TX path loss module value is a 2's complement.**

Parameter Index	Param Data length	mParamData	
		0	

TX_POWER_GAIN_K	2	0x03 (Set)	TX path loss module Value
-----------------	---	------------	---------------------------

The settings APIs define below:

```
pParam->ParameterIndex = TX_PATH_LOSS_MODULE;
pParam->mParamData[0] = 3; //set value
pParam->mParamData[1] = TX_path_loss_module Value; //set value
pBluetoothModule->UpDataParameter(pBluetoothModule,pParam);
if (pBluetoothModule->ActionControlExcute(pBluetoothModule) !=0){
    goto error;
}
```

### 6.1.5 Set TX Power Level To RAM

For verify TX performance need to setting temporarily set the current TX level. You can use the APIs below to set the device to the ram.

```
/*-----*/
#define HCI_VENDOR_SETTING_TX_LEVEL_TO_RAM 0xFCE7
unsigned char powerlevel[512]={0};
unsigned char pEvtBuf[512] = {0};

String text = ComboBox_MaxTXPowerofLegacy1M->Text.SubString(1,4);
powerlevel[0] = TX_level_1M;
powerlevel[1] = TX_level_2M;
powerlevel[2] = TX_level_3M;
powerlevel[3] = TX_level_BLE1M;
powerlevel[4] = TX_level_BLE2M;

if(pBluetoothModule->SendHciCommandWithEvent(pBluetoothModule,HCI_VENDO
R_SETTING_TX_LEVEL_TO_RAM , LEN_5_BYTE, powerlevel, 0x0E, pEvtBuf))
{
    printf(">>Set TX Power Level fail");
}
else
{
    printf(">>Set TX Power Set TX Power Level successfully");
}
/*-----*/
```

### 6.1.6 Set Crystal cap vale

If need to set crystal cap then use this function to set it.

```
pParam->ParameterIndex = SET_XTAL;
pParam->Rtl8761Xtal = InitIndex(0x20);
```

```
pBluetoothModule->UpDataParameter(pBluetoothModule,pParam);  
pBluetoothModule->ActionControlExcute(pBluetoothModule);
```

Realtek

## 6.2 Bluetooth Calibrates Thermal (Tmeter) vaule by MP

Normal driver will load thermal meter to do power tracking. So this value must be filled on correct eFuse location. Use the command below to read Bluetooth thermal meter once. If platform is Windows, please through BT DLL of Realtek to obtain raw thermal value. Please refer below sample code of use DLL:

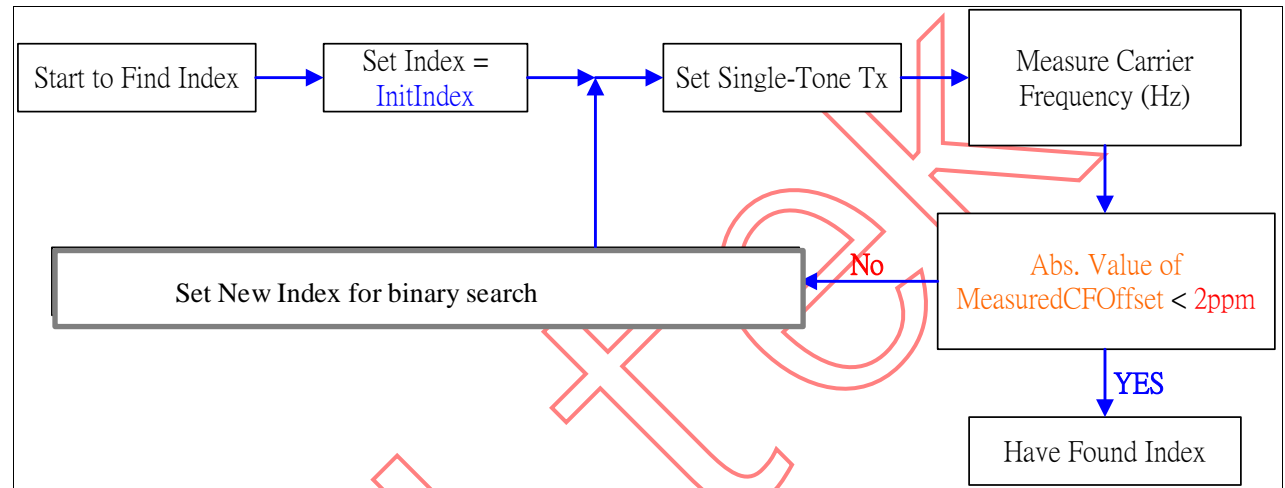
```
/*-----*/  
  
BT_DEVICE_REPORT *pModuleBtReport=&mBT_DEVICE_REPORT;  
//Read Tmeter  
pBluetoothModule->ActionReport(pBluetoothModule,REPORT_THERMAL,pModuleB  
tReport);  
{  
    printf(">>Get Thermal is Fail...!!\n");  
    return;  
}  
printf(">>Thermal=(%d)\n",pModuleBtReport->CurrThermalValue );  
  
/*-----*/
```

Note: The Thermal Value is RAW Data of Realtek define.

## 6.3 Bluetooth Calibrates Crystal (Xtal) Cap by Non-Signaling mode

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. The steps to PG efuse is in section 6.3.

The flow of calibration Bluetooth step is as below:



### 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 2.441GHz band is about 10KHz(±5KHz).

Step1 : Set Index = **InitIndex**(0x20)

```

pParam->ParameterIndex = SET_XTAL;
pParam->Rtl8761Xtal = InitIndex(0x20);
pBluetoothModule->UpDataParameter(pBluetoothModule,pParam);
pBluetoothModule->ActionControlExcute(pBluetoothModule);
  
```

Step2: Set Single-Tone Tx

```

pBtParam ->mChannelNumber=39;
pBtParam ->mPacketType=3DH5;
pBtParam ->mPayloadType=ALL0;
pBtParam ->mTxPacketCount=0(Infinite);
pBtParam ->mWhiteningCoeffValue =0xFF(disable);
pBtParam ->mTxGainIndex=7;
pBtParam ->mPacketHeader=0XA078;
pBtParam ->mHitTarget= 0x000000c6967e
pBtParam ->ParameterIndex= CONTINUE_TX_START;
UpDataParameter(pBtModule,pBtParam);
ActionControlExcute(pBtModule);
  
```



Step 3: Stop Single-Tone Tx.

```
ParameterIndex= CONTINUE_TX_STOP;  
UpDataParameter(pBtModule,pBtParam);  
ActionControlExcute(pBtModule);
```

Step 4: Go to Step1 and re-tune Index value until find the best Crystal index value.

Step 5: Write Crystal value to Efuse. (Please refer to Chapter 6.3)

*Note: The method of calibration depends on different chips.*

Realtek

## 6.4 Verify Bluetooth Legacy Tx Performance by Non-Signaling mode

To measure the DUT Bluetooth legacy TX power/initial Carrier offset/modulation characteristics to check Tx performance is ok or not. **The DUT parameter setting is determined by the pattern of the instrument.**

Bluetooth legacy TX criterion is shown as below:

	Test Item	Sub Test Item	Packet Type	Channel	Criterion
					Bluetooth Spec.
Verify Tx DH1	Maximum Output Power	Average Power	DH1	Low (CH6-2408MHZ)	> 0dBm
				Middle (CH42-2444MHZ)	> 0dBm
				High (CH70-2472MHZ)	> 0dBm
	Modulation Characteristics	Delta F1 Avg.	DH1	Low (CH6-2408MHZ)	140KHz ~ 175KHz
		Delta F2 Max.		Middle (CH42-2444MHZ)	> 115KHz
		Modulation Index		High (CH70-2472MHZ)	> 0.8
	Initial Carrier Frequency Error		DH1	Low (CH6-2408MHZ)	-20KHz ~ 20KHz
				Middle (CH42-2444MHZ)	
				High (CH70-2472MHZ)	
Verify Tx 3DH1	Maximum Output Power	Average Power	3DH1	Low (CH6-2408MHZ)	> 0dBm
				Middle (CH42-2444MHZ)	> 0dBm
				High (CH70-2472MHZ)	> 0dBm
	Modulation Characteristics	RMS DEVM	3DH1	Low (CH6-2408MHZ)	0.13
		Peak DEVM		Middle (CH42-2444MHZ)	0.25
		99% DEVM		High (CH70-2472MHZ)	0.20
	Initial Carrier Frequency Error		3DH1	All	-20KHz ~ 20KHz

**Table The recommended test items of Bluetooth Tx**

For example, use adb commands android platform, device is UART interface.

step by step command:

- (1) Enter MP Mode and download patch code
- (2) Set Parameter :

Test Item		adb command
	Test Item	Channel = 6
DH1	Maximum Power	mChannelNumber=0x06; mPacketType=0x00; mPayloadType=0x07; mTxPacketCount=0x00; mWhiteningCoeffValue=0x80; mTxGainIndex=0x7; mPacketHeader=0x33820; mHitTarget=0x0000009e8b33
DH1	Delta F1	mChannelNumber=0x06; mPacketType=0x00; mPayloadType=0x05; mTxPacketCount=0x00; mWhiteningCoeffValue=0x80; mTxGainIndex=0x7; mPacketHeader=0x33820; mHitTarget=0x0000009e8b33
DH1	Delta F2	mChannelNumber=0x06; mPacketType=0x00; mPayloadType=0x02; mTxPacketCount=0x00; mWhiteningCoeffValue=0x80; mTxGainIndex=0x7; mPacketHeader=0x33820; mHitTarget=0x0000009e8b33
3DH1	ALL	mChannelNumber=0x06; mPacketType=0x06; mPayloadType=0x07; mTxPacketCount=0x00; mWhiteningCoeffValue=0x80; mTxGainIndex=0x7; mPacketHeader=0x15C40; mHitTarget=0x0000009e8b33

Test Item		adb command
	Test Item	Channel = 70
DH1	Maximum Power	mChannelNumber=0x46; mPacketType=0x00; mPayloadType=0x07; mTxPacketCount=0x00; mWhiteningCoeffValue=0x80; mTxGainIndex=0x7; mPacketHeader=0x33820;

		mHitTarget=0x0000009e8b33
DH1	Delta F1	mChannelNumber=0x46; mPacketType=0x00; mPayloadType=0x05; mTxPacketCount=0x00; mWhiteningCoeffValue=0x80; mTxGainIndex=0x7; mPacketHeader=0x33820; mHitTarget=0x0000009e8b33
DH1	Delta F2	mChannelNumber=0x46; mPacketType=0x00; mPayloadType=0x02; mTxPacketCount=0x00; mWhiteningCoeffValue=0x80; mTxGainIndex=0x7; mPacketHeader=0x33820; mHitTarget=0x0000009e8b33
3DH1	ALL	mChannelNumber=0x46; mPacketType=0x06; mPayloadType=0x07; mTxPacketCount=0x00; mWhiteningCoeffValue=0x80; mTxGainIndex=0x7; mPacketHeader=0x15C40; mHitTarget=0x0000009e8b33

(3)Run Packe Tx

(4) measured by Bluetooth test instrument(ex. Letepoint IQNxN)

PS. "ActionReport" should be called every 200ms.

(5)Stop Packet Tx

PS. If you need to test other parameters, please stop packet tx and go back to step 2

## 6.5 Verify Bluetooth Legacy Rx Performance by Non-Signaling mode

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 IQNxn). Bluetooth Rx criterion is shown as below:

Verify Bluetooth Rx	Test Item	Packet Type	Criterion
			Bluetooth Spec
	Sensitivity	DH1 or 3DH5	<b>&lt; -70dBm</b>

For final MP, Rx can just test DH1, 3DH5 BER at sensitivity criterion power level at channel 0 and 78 to reduce time. All Bluetooth Rx criterion is shown as:

Test Item		Criterion(Bluetooth Spec)
Channel	Packet type	< -70dBm
6	DH1	< -70dBm
42	DH1	< -70dBm
70	DH1	< -70dBm
6	3DH1	< -70dBm
42	3DH1	< -70dBm
70	3DH1	< -70dBm

**Figure The recommended test items of Bluetooth Rx**

(1) Enter MP Mode and download patch code

(2) Set Parameter :

Test Item		adb command
Channel	Packet type	
6	DH1	mChannelNumber=0x06; mPacketType=0x00; mPayloadType=0x07; mWhiteningCoeffValue=0x80; mPacketHeader=0x33820; mHitTarget=0x000000c6967e
42	DH1	mChannelNumber=0x2a; mPacketType=0x00; mPayloadType=0x07; mWhiteningCoeffValue=0x80; mPacketHeader=0x33820; mHitTarget=0x000000c6967e
70	DH1	mChannelNumber=0x46; mPacketType=0x00;

		mPayloadType=0x07; mWhiteningCoeffValue=0x80; mPacketHeader=0x33820; mHitTarget=0x000000c6967e
6	3DH1	mChannelNumber=0x06; mPacketType=0x06; mPayloadType=0x07; mWhiteningCoeffValue=0x80; mPacketHeader=0x33820; mHitTarget=0x000000c6967e
42	3DH1	mChannelNumber=0x2a; mPacketType=0x06; mPayloadType=0x07; mWhiteningCoeffValue=0x80; mPacketHeader=0x15C40; mHitTarget=0x000000c6967e
70	3DH1	mChannelNumber=0x46; mPacketType=0x06; mPayloadType=0x07; mWhiteningCoeffValue=0x80; mPacketHeader=0x15C40; mHitTarget=0x000000c6967e

(3)To setting Parameter with the Bluetooth test instrument. Bluetooth test instrument begin transmit..

(4)Run Packet Rx

(5) Report Received Result.

PS. "ActionReport" should be called every 200ms.

(6)Stop Packet Rx

PS. If you need to test other parameters, please stop packet Rx and go back to step 2.

## 6.6 Verify Bluetooth BLE4.0 Tx Performance

To measure the DUT BLE TX power and modulation index to check BLE TX performance is ok or not. Bluetooth BLE TX criterion is shown as below:

	Test Item	Sub Test Item	Payload Type	Channel	Criterion
					Bluetooth Spec.
Verify BLE Tx	BLE Output Power	Average Power	PRBS9	Low (CH0-2402MHZ)	> 0dBm
				Middle (CH19-2440MHZ)	> 0dBm
				High (CH34-2470MHZ)	> 0dBm
	Modulation Characteristics	Delta F1 Avg.	BT_PAYLOAD_TYPE_1111_0000	2440MHZ	225 kHz ~ 275 kHz
		Delta F2 Max.	BT_PAYLOAD_TYPE_1010	CH19-	≥ 185 kHz
		Modulation Index	None		≥ 0.8

**Table The recommended test items of Bluetooth BLE Tx**

For example, use adb commands android platform, device is UART interface.

step by step command:

- (1) Enter MP Mode and download patch code
- (2) Set Parameter :

Test Item		adb command
	Test Item	Channel = 6
BLE	Average Power	mChannelNumber=0x00; //(change channel :0x19,0x34) mPayloadType=0x00; //Payload type = PRBS9 mTxGainIndex=0x6;
BLE	Delta F1	mChannelNumber=0x00; //(change channel :0x19,0x34) mPayloadType=0x01; //Payload type = 11110000 mTxGainIndex=0x6;
BLE	Delta F2	mChannelNumber=0x00; //(change channel :0x19,0x34) mPayloadType=0x02; //Payload type = 10101010 mTxGainIndex=0x6;

## 6.7 Verify Bluetooth BLE4.0 Rx Performance

Measure the DUT BLE Rx sensitivity to check Rx performance is ok or not. The Bluetooth Rx criterion is shown as below:

Channel	Payload Type	BLE sensitivity limit	Criterion Bluetooth Spec
Low (CH0-2402MHZ)	PRBS9	PER<= 30.800 %	< -70dBm
Middle (CH19-2440MHZ)			
High (CH34-2470MHZ)			

**Figure The recommended test items of Bluetooth BLE Rx**

To calculate the Packet Error Rate(PER)

$$\text{PER}\% = 100 * (1 - (\text{Packets Received} / \text{Packets Send}))$$

The step by step list below :

(1) Enter MP Mode and download patch code

(2) Set Parameter :

Test Item		adb command
Item	Payload type	
Sensitivity	PRBS9	mChannelNumber=0x00; (change channel :0x19/0x34) mPayloadType=0x00; //PRBS9

(3) To setting Parameter with the Bluetooth test instrument.

(4) Run LE Packet Rx

(5) Stop LE Packet Rx and to obtain the receive packet count

(6) Calculate PER

If ( PER < limit ) , goto step (3) , else Stop



## 6.8 Verify Bluetooth BLE5.0 Tx Performance

To measure the DUT BLE 5.0 TX power and modulation index to check BLE TX performance is ok or not. Bluetooth BLE 5.0 TX criterion is shown as below. **For vender's TX verification, please adapt target power value as your own spec.** Bluetooth Target tx power spec depends on different chips. Bluetooth BLE 5.0 TX criterion is shown as below:

PHY Spec	Test Item	Sub Test Item	Payload Type	Channel	Criterion Bluetooth Spec.
2M	BLE Output Power	Average Power	PRBS9	0/3/19/22/36/39	2.5 dBm ~ 6.5 dBm NOTE: Max variation = Target power value +/- 2 dB*
LR S2					
LR S8					
2M	Carrier freq. offset & drift	None	PRBS9		2M: < 20 kHz
LR S2			PRBS9		S8: < 19.2 kHz
LR S8			PRBS9		
2M	Modulation Characteristic	Delta F1 Avg.	BT_PAYLOAD_TY PE_1111_0000		2M: 450 ~ 550 kHz S8: 225 ~ 275 kHz
LR S2		Delta F2 Max.	BT_PAYLOAD_TY PE_1010		2M: > 370 kHz
LR S8		F2avg/F1avg	None		2M: > 0.8

Table 1 Bluetooth BLE 5.0 TX criterion

**\*Max variation spec only available after doing TX calibration flow**

For this test must to setting current tx power by Calibrates TX power and

For example, the step by step :

(1) Set Parameter :

Test Item	Channel = 0/3/19/22/36/39
Average Power	<p>pParam -&gt;ParameterIndex =LE_TX_DUT_TEST_CMD</p> <p>pParam -&gt;mChannelNumber=0x00;//channel</p> <p>pParam -&gt;mPayloadType=0x00; //Payload type = PRBS9</p> <p>pParam-&gt;PHY = 2 ; // 1:LE 1M 2:LE 2M 3: LE Coded PHY</p> <p>pParam-&gt;ModulationIndex = 1;</p> <p>// 0: standard modulation 1: stable modulation</p>
Delta F1	<p>pParam -&gt;ParameterIndex =LE_TX_DUT_TEST_CMD</p> <p>pParam -&gt;mChannelNumber=0x00;//channel</p> <p>pParam -&gt;mPayloadType=0x01; //Payload type =00001111</p> <p>pParam-&gt;PHY = 2 ; // 1:LE 1M 2:LE 2M 3: LE Coded PHY</p> <p>pParam-&gt;ModulationIndex = 1;</p> <p>// 0: standard modulation 1: stable modulation</p>

Delta F2	<p>pParam -&gt;ParameterIndex =LE_TX_DUT_TEST_CMD</p> <p>pParam -&gt;mChannelNumber=0x00;//channel</p> <p>pParam -&gt;mPayloadType=0x02; //Payload type =10101010</p> <p>pParam-&gt;PHY = 2 ; // 1:LE 1M 2:LE 2M 3: LE Coded PHY</p> <p>pParam-&gt;ModulationIndex = 1;</p> <p>// 0: standard modulation 1: stable modulation</p>
----------	---

pBluetoothModule->UpDataParameter(pBluetoothModule,pParam);

(2)Execute active job - Run LE DUT TX

pBluetoothModule->ActionControlExcute(pBluetoothModule);

(3) Measured by Bluetooth test instrument(ex. Letepoint IQNxN)

(4) Execute active job - Run Packet Stop

Ex.

ParameterIndex = FW\_PACKET\_TX\_STOP

pBluetoothModule->ActionControlExcute(pBluetoothModule);

## 6.9 Verify Bluetooth BLE5.0 Rx Performance

To calculate the Packet Error Rate(PER)

**PER%= 100\*(1-(Packets Received /Packets Send)**

Measure the DUT BLE Rx sensitivity to check Rx performance is ok or not. The Bluetooth Rx criterion is shown as below:

Channel	Spec	Payload Type	BLE sensitivity limit	Criterion
	PHY Spec			Bluetooth Spec
0/3/19/22/36/39	1M	PRBS9	PER<= 30.800 %	< -70 dBm
	2M	PRBS9	PER<= 30.800 %	< -70 dBm
	LRS2	PRBS9	PER<= 30.800 %	< -75 dBm
	LRS8	PRBS9	PER<= 30.800 %	< -82 dBm

Table 2 The recommended test items of Bluetooth BLE 5.0 Rx

The simple sample code of RF dll show as:

The step by step list below :

- (1) Enter Hci reset to reset the DUT.
- (2) Set Parameter & Run LE Packet Rx

The LE RX sample code below:

```
pParam->ParameterIndex=LE_RX_DUT_TEST_CMD; //23
pParam->mChannelNumber= 0; //channel :0~39
pParam->PHY= 2; // 1:LE 1M 2:LE 2M 3: LE Coded PHY
pParam->ModulationIndex = 1; // 0: standard modulation 1: stable modulation
pBluetoothModule->UpDataParameter(pBluetoothModule,pParam);
pBluetoothModule->ActionControlExcute(pBluetoothModule
```

- (3) To setting Parameter with the Bluetooth test instrument ,then start TX until the end.
- (4) Stop LE Packet Rx and to obtain the receive packet count

```
pParam->ParameterIndex = LE_DUT_TEST_END_CMD; //23
pBluetoothModule->UpDataParameter(pBluetoothModule,pParam);
pBluetoothModule->ActionControlExcute(pBluetoothModule
```

- (1) Calculate PER:

If ( PER< limit) , goto step (2) , else to To finish.

## 7. Write EFUSE Data to Bluetooth Device

The EFUSE data is composited in entry unit and little endian format, which is 2-byte offset, 1-byte length and data in length size.

The EFUSE write/read operations are supported by the below series of Bluetooth chips.

Realtek Bluetooth Chip Number
RTL8761A series

### 7.1 Write BT MAC Address to EFUSE

The EFUSE data of Bluetooth MAC address is 6-byte length, and offset started at 0x3C. The Bluetooth MAC address defined condition:

- 1) Byte0 and Byte 1 can't be 0xFF at the same time.
- 2) Byte2 and Byte 3 can't be 0xFF at the same time.
- 3) Byte4 and Byte5 can't be 0xFF at the same time.

Byte Number	5	4	3	2	1	0
Range	0~0xFFFE		0~0xFFFE		0~0xFFFE	

If you want to use 0xFFFF as Bluetooth MAC address, you must first write a non-0xFFFF value and then write 0xFFFF.

For writed to efuse example, if the default value of Bluetooth MAC address is 0x00E04C829987 (00:E0:4C:82:99:87), and the user wants to update the value to 0x00E023345678 (00:E0:23:34:56:78) like below.

Offset	Default value	Example value	Description
0x3c-0x41	0x00E04C829987	0x00E023345678	BD_ADDR

The data array is organized as below: item index should be 0x00, and sub-index should be fixed as 0x01.

Byte Numb	0	1	2	3	4	5	6	7	8	9
Description	Sub-Index	Offset (Low)	Offset (High)	Length	BD_ADDR[0]	BD_ADDR[1]	BD_ADDR[2]	BD_ADDR[3]	BD_ADDR[4]	BD_ADDR[5]
Array	0x01	0x3C	0x00	0x06	0x78	0x56	0x34	0x23	0xE0	0x00

Use the command below to write MAC address to EFUSE.

```
pParam->ParameterIndex = WRITE_EFUSE_DATA;
pPGRawData[0] = 0x01;
pPGRawData[1] = 0x3C;
pPGRawData[2] = 0x00;
pPGRawData[3] = 0x06;
pPGRawData[4] = 0x78;
pPGRawData[5] = 0x56;
pPGRawData[6] = 0x34;
pPGRawData[7] = 0x23;
pPGRawData[8] = 0xE0;
pPGRawData[9] = 0x00;

pBluetoothModule->UpDataParameter(pBluetoothModule,pParam);
pBluetoothModule->ActionControlExcute(pBluetoothModule) ;
```

It is recommended that MAC address should be verified by the below commands.

```
pParam->ParameterIndex = READ_EFUSE_DATA;
pPGRawData[0] = 0x01; // BT Efuse
pPGRawData[1] = 0x3c;
pPGRawData[2] = 0x00;
pPGRawData[3] = 0x06;
pBluetoothModule->UpDataParameter(pBluetoothModule,pParam);
pBluetoothModule->ActionControlExcute(pBluetoothModule);

pBluetoothModule->ActionReport(pBluetoothModule,
REPORT_LOGICAL_EFUSE, &mBT_DEVICE_REPORT);
//Get BT addr = pReport[9], pReport[8],..., pReport[5], pReport[4]);
```

## 7.2 Write Power On Function to EFUSE

The EFUSE data of Power On function has two entries to configure: offset 0x0177 is used to set Power On function enabled or disabled, while offset 0x30-0x31 is used to set Power On level.

Offset	Default value	Example value	Description
0x30-0x31	0x5240	0x5052	If 0x30[2:1] == 0x01 && 0x30[4] == 0x01, then BT_WAKE_HOST pin is set to high by default; otherwise low.
0x0177	0x3C	0x3D	0x177[0]: enable power on

The data array is organized as below: item index should be 0x00, and sub-index should be fixed as 0x01.

Byte Number	0	1	2	3	4	5
Description	Sub-Index	Offset (Low)	Offset (High)	Length	Power On Enable Power On Level (Low)	Power On Level (High)
Power on Enable Array	0x01	0x77	0x01	0x01	0x3C[0x3D]	
Power on Level Array	0x01	0x30	0x00	0x02	0x40[0x52]	0x52[0x50]

Use the command below to disable power on function.

```
pParam->ParameterIndex = WRITE_EFUSE_DATA;
pPGRawData[0] = 0x01;
pPGRawData[1] = 0x77;
pPGRawData[2] = 0x01;
pPGRawData[3] = 0x01;
pPGRawData[4] = 0x3C;
pBluetoothModule->UpDataParameter(pBluetoothModule,pParam);
pBluetoothModule->ActionControlExcute(pBluetoothModule) ;
```

Use the command below to enable power on function.

```
pParam->ParameterIndex = WRITE_EFUSE_DATA;
pPGRawData[0] = 0x01;
pPGRawData[1] = 0x77;
pPGRawData[2] = 0x01;
pPGRawData[3] = 0x01;
pPGRawData[4] = 0x3D;
pBluetoothModule->UpDataParameter(pBluetoothModule,pParam);
pBluetoothModule->ActionControlExcute(pBluetoothModule) ;
```

Use the command below to set power on active level high (keep low level when idle).

```
pParam->ParameterIndex = WRITE_EFUSE_DATA;
pPGRawData[0] = 0x01;
pPGRawData[1] = 0x30;
pPGRawData[2] = 0x00;
```

```

pPGRawData[3] = 0x02;
pPGRawData[4] = 0x40;
pPGRawData[5] = 0x52;
pBluetoothModule->UpDataParameter(pBluetoothModule,pParam);
pBluetoothModule->ActionControlExcute(pBluetoothModule) ;

```

Use the command below to set power on active level low (keep high level when idle).

```

pParam->ParameterIndex = WRITE_EFUSE_DATA;
pPGRawData[0] = 0x01;
pPGRawData[1] = 0x30;
pPGRawData[2] = 0x00;
pPGRawData[3] = 0x02;
pPGRawData[4] = 0x52;
pPGRawData[5] = 0x50;
pBluetoothModule->UpDataParameter(pBluetoothModule,pParam);
pBluetoothModule->ActionControlExcute(pBluetoothModule) ;

```

### 7.3 Write Crystal Cap Calibration Value to EFUSE

Normal driver will load the Crystal Cap value in the initial stage, so it must be well-calibrated and written to EFUSE.

The EFUSE data of Crystal Cap Calibration is 1-byte length, and offset started at 0x01E6.

Offset	Default value	Example value	Description
0x01E6	0x20	0x00~0x3F	XTAL_K Value Bit[5:0], Xi, Xo Range

The data array is organized as below: item index should be 0x00, and sub-index should be fixed as 0x01.

Byte Number	0	1	2	3	4
Description	Sub-Index	Offset (Low)	Offset (High)	Length	Xtal value
Crystal Cap Calibration Array	0x01	0xE6	0x01	0x01	0~63

Use the command to write Crystal (Xtal) value 0x20 to EFUSE.

```

pParam->ParameterIndex = WRITE_EFUSE_DATA;
pPGRawData[0] = 0x01;
pPGRawData[1] = 0xE6;
pPGRawData[2] = 0x01;
pPGRawData[3] = 0x01;
pPGRawData[4] = 0x20;
pBluetoothModule->UpDataParameter(pBluetoothModule,pParam);
pBluetoothModule->ActionControlExcute(pBluetoothModule) ;

```

It is recommended that Crystal Cap value should be verified by the below commands.

```

pParam->ParameterIndex = READ_EFUSE_DATA;
pPGRawData[0] = 0x01; // BT Efuse
pPGRawData[1] = 0xE6;
pPGRawData[2] = 0x01;
pPGRawData[3] = 0x01;
pBluetoothModule->UpDataParameter(pBluetoothModule,pParam);
pBluetoothModule->ActionControlExcute(pBluetoothModule);

pBluetoothModule->ActionReport(pBluetoothModule,
REPORT_LOGICAL_EFUSE, &mBT_DEVICE_REPORT);
//Get Crystal (Xtal) value = pReport[4];

```

## 7.4 Write VID (Vendor ID) and PID (Product ID) to EFUSE

The VID and PID configuration is used only for USB interface chips, and defined in system EFUSE.

Offset	Default value	Example value	Description
0x24	0xDA	0xDA	USB Vendor ID[7:0]
0x25	0x0B	0x0B	USB Vendor ID[15:8]
0x26	0x50	0x61	USB Product ID[7:0]
0x27	0x28	0x87	USB Product ID[15:8]

The data array is organized as below: item index should be 0x00, and sub-index should be fixed as 0x02.

Byte Number	0	1	2	3	4	5	6	7
Description	Sub-Index	Offset (Low)	Offset (High)	Length	VID LSB	VID MSB	PID LSB	PID MSB



VID & PID Array	0x02	0x24	0x00	0x04	0xDA	0x0B	0x61	0x81

Use the command below to write VID & PID to EFUSE.

```

pParam->ParameterIndex = WRITE_EFUSE_DATA;
pPGRawData[0] = 0x02;
pPGRawData[1] = 0x24;
pPGRawData[2] = 0x00;
pPGRawData[3] = 0x04;
pPGRawData[4] = 0xDA;
pPGRawData[5] = 0x0B;
pPGRawData[6] = 0x61;
pPGRawData[7] = 0x81;
pBluetoothModule->UpDataParameter(pBluetoothModule,pParam);
pBluetoothModule->ActionControlExcute(pBluetoothModule) ;

```