

© 2013 Realtek Semiconductor Corp. All rights reserved. No part of this document may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language in any form or by any means without the written permission of RealTEK Semiconductor Corp.

Realtek Bluetooth Mass Production DLL Specification

Change History

Version	Editor	Date	Remarks
v1.0	Bryant	2011/07/26	First Release.
v2.0	Bryant	2012/10/17	1.Add host control interface

REALTEK Confidential

Table of Contents:

Change History	2
Table of Contents:	3
1. General Functions.....	5
1.1. RTKBT_OpenAdapter	5
1.2. RTKBT_CloseAdapter	6
1.3. RTKBT_GetLastError	7
1.4. RTKBT_ShowUI.....	8
1.5. RTKBT_HCIReset	9
1.6. RTKBT_BTTestMode.....	10
1.7. RTKBT_SetTRxParameters	11
1.8. RTKBT_StartPacketTx	14
1.9. RTKBT_StartContinueTx	15
1.10. RTKBT_StartRx.....	17
1.11. RTKBT_StopTest.....	18
1.12. RTKBT_QueryTRxStatus	20
1.13. RTKBT_GetStatistics.....	21
1.14. RTKBT_ClearCounter	22
1.15. RTKBT_TxPowerAdjust.....	23
1.16. RTKBT_ReadThermalMeter.....	24
1.17. RTKBT_WriteReg	25
1.18. RTKBT_ReadReg	26
1.19. RTKBT_SendHCICmd	27
2. Appendix	29
2.1. Error Code Descriptions	29
2.2. Example.....	31
2.2.1. BT DUT Test Mode (Link Mode Test), Host Interface: BT-USB	32
2.2.2. Packet-Tx Test (Non-Link Mode Test), Host Interface: WiFi-PCIe.....	33
2.2.3. Packet-Rx Test (Non-Link Mode Test), Host Interface: Android ADB	35

Table 1. DLL API Name Table

Direction	API Name	Remark
General		
	RTKBT_OpenAdapter	
	RTKBT_CloseAdapter	
	RTKBT_GetLastError	
	RTKBT_ShowUI	
	RTKBT_HCIReset	
	RTKBT_BTTestMode	
	RTKBT_SetTRxParameters	
	RTKBT_StartPacketTx	
	RTKBT_StartContinueTx	
	RTKBT_StartRx	
	RTKBT_StopTest	
	RTKBT_QueryTRxStatus	
	RTKBT_GetStatistics	
	RTKBT_ClearCounter	
	RTKBT_TxPowerAdjust	
	RTKBT_ReadThermalMeter	
	RTKBT_WriteReg	
	RTKBT_ReadReg	
	RTKBT_SendHCICmd	

1. General Functions

1.1. RTKBT_OpenAdapter

bool RTKBT_OpenAdapter(UINT8 HostType, UINT8 PortNo, UINT32 Rate);

Parameters

HostType

The host interface type. 0: USB, 1:UART, 2:PCle,3: reserved, 4:WiFi USB, 5:WiFi SDIO, 6:WiFi PCle, 7:Linux by ADB, 8:Linux by UART

PortNo

The port number for specified host interface. The valid port number is 1~2 for USB (HostType =0) & PCle (HostType =2), and 1~12 for UART (HostType =1) & Linux by UART (HostType =8).

Rate

The baud rate in kbps for UART interface. The allowable value is 9600, 14400, 19200, 38400, 57600, 115200. The default value is 115200. For other interfaces, this parameter is not required and should be set to 0.

Return Values

If the operation completes successfully, the return value is TRUE. If the operation fail , the return value is FALSE and the application can call [RTKBT_GetLastError\(\)](#) to retrieve the error code.

Remarks

This function should be called prior to other functions exported in this DLL

Example

```
//open Port1 USB as HCI interface

if (!RTKBT_OpenAdapter(0,1,0))
    printf("Open Adapter Error = %d", RTKBT_GetLastError());
else
    printf("Initial Adapter ok !!");
```

1.2. RTKBT_CloseAdapter

bool RTKBT_CloseAdapter();

Parameters

None

Return Values

If the operation completes successfully, the return value is TRUE. If the operation fails, the return value is FALSE and the application can call [RTKBT_GetLastError\(\)](#) to retrieve the error code.

Remarks

This function should be called before exit the DLL. The application can still reinitial the adapter by calling the [RTKBT_OpenAdapter\(\)](#) after executing the [RTKBT_CloseAdapter\(\)](#).

If Tx/Rx test is running in link mode, call [RTKBT_HCIReset\(\)](#) first to stop testing before executing this function.

If Tx/Rx test is running in non-link mode, call [RTKBT_StopTest\(\)](#) first to stop testing before executing this function.

Example

```
//Close Port1 USB as HCI interface
```

```
if (!RTKBT_CloseAdapter())  
    printf("Close Adapter Error = %d", RTKBT_GetLastError());  
else  
    printf("Close Adapter ok !!");
```

1.3. RTKBT_GetLastError

bool RTKBT_GetLastError();

Parameters

None

Return Values

Return the error code that generated by the last calling DLL function. The error code is defined in the mp8723xBT.h file. The error code 0 indicates no error, and the descriptions of other non-zero error codes are listed in the [appendix](#).

Remarks

This function should be called if the application needs to check the error message.

Example

See the example code of the [RTKBT_CloseAdapter\(\)](#).

1.4. RTKBT_ShowUI

bool RTKBT_ShowUI(bool IsShow);

Parameters

IsShow

True: show GUI, False: hide GUI

Return Values

If the operation completes successfully, the return value is TRUE. If the operation fails, the return value is FALSE and the application can call [RTKBT_GetLastError\(\)](#) to retrieve the error code.

Remarks

This function is used for debug. The DLL is built from an GUI application, and all the DLL functions is correspond to an GUI action. Therefore, the user application can display the DLL GUI to check the real Bluetooth operation status.

Example

```
if (!RTKBT_ShowUI(true))
    printf("Show DLL UI Error = %d", RTKBT_GetLastError());
else
    printf("Show DLL UI ok !!");
```


1.5. RTKBT_HCIReset

bool RTKBT_HCIReset();

Parameters

None

Return Values

If the operation completes successfully, the return value is TRUE. If the operation fail , the return value is FALSE and the application can call [RTKBT_GetLastError\(\)](#) to retrieve the error code.

Remarks

This function issues an HCI_Reset and waits for the appropriate command complete

Example

```
//Issue an HCI Reset  
  
if (!RTKBT_HCIReset())  
    printf("HCI Reset Error = %d", RTKBT_GetLastError());  
else  
    printf("HCI Reset ok !!");
```

1.6. RTKBT_BTTestMode

bool RTKBT_BTTestMode(UINT8 Type);

Parameters

HostType

The test mode type: 0: Bluetooth DUT Test Mode

Return Values

If the operation completes successfully, the return value is TRUE. If the operation fails, the return value is FALSE and the application can call [RTKBT_GetLastError\(\)](#) to retrieve the error code.

Remarks

If the test mode type is "Bluetooth DUT Test Mode", this function performs two steps:

1. Implements the command HCI_WriteScan_Enable with the following parameter:
0x03 - Inquiry scan enabled, Page scan enabled.
2. Implements the command HCI_Enable_Device_Under_Test_Mode.

After BT controller goes into DUT test mode successfully, it will accept connect request automatically if BT tester wants to create connection with it. Therefore, the BT tester can perform Signaling-Test that defined in the Bluetooth spec.

The application should call [RTKBT_HCIReset\(\)](#) if it wants to exit BT DUT test mode.

Example

```
if (!RTKBT_BTTestMode(0))
{
    printf("BT Test Mode Error = %d", RTKBT_GetLastError());
    return;
}
else
{
    printf("BT DUT Test Mode ok !!");
    // Place BT Tester control code here to create connection with DUT
}
```

1.7. RTKBT_SetTRxParameters

bool RtkBTAPI RTKBT_SetTRxParameters(RTKBT_TRXPARAM params);

Parameters

Params is a RTKBT_TRXPARAM structure defined in the header file: mp8723xBT.h, it includes the following members:

Rate

Value range: 1~4. 1: 1M, 2: 2M, 3: 3M, 4: BT4.0 LE. It specifies the modulation type of the packet for "Packet Tx" and "Continue Tx" test actions, and the desired rate of the received signal for "Packet Rx" test action.

Channel

The operation channel, value range: 0~78. The valid value range is 0~39 for BT 4.0 LE test (Params Rate = 4)

TxPktCnt

Value range: 0~0xffffffff. If the value is "0", the packet would not stop until *RTKBT_StopTest()* is called. This parameters is only effective in the test action "Packet Tx", it should be set to 0 for other tests. For "Bluetooth DUT Test Mode", the packet would not stop until *RTKBT_StopTest()* is called.

TxPktInterval

Value range: 0~15. The real Tx packet interval is (value+1)*100us. This parameters is only effective in the test action "Packet Tx", and it should be set to 0 for other test.

PayloadType

For RTL8723a:

BR&EDR "Packet-Rx" and "Continue Tx", value range: 0~4.

0: All 0's, 1: All 1's, 2: 0101, 3: 1010, 4: 0x0~0xf.

BR&EDR "Packet-Tx", value range: 0~4.

0: 01010101, 1: All 1's, 2: All 0's, 3: 11110000, 4: Normal Data.

BT4.0 LE test "Packet Tx" and "Packet Rx", value range: 0~4.

0: PRBS9, 1: 11110000, 2: 10101010, 3: All 1's, 4: All 0's.

For other RTK BT chip:

BR&EDR "Packet Tx", "Packet-Rx" and "Continue Tx", value range: 0~7.

0: All 0's, 1: All 1's, 2: 0101, 3: 1010, 4: 0x0~0xf, 5: 00001111
6: 11110000, 7: PRBS9.

BT4.0 LE test "Packet Tx" and "Packet Rx", value range: 0~4.

0: PRBS9, 1: 11110000, 2: 10101010, 3: All 1's, 4: All 0's.

PayloadLength

BR & EDR → value range: 1~8191, unit: bits. LE → value range: 1~37, unit: bytes.

(Note: For test action "Packet Rx", the *PayloadLength* is used for BER calculation.)

PacketHeader

Value range: 0x00000~0x3ffff, the packet header length must be 18 bits. This parameter is only effective in the test actions: "Packet Tx" and "Packet Rx", it should be set to 0x3ffff for other tests.

WhitenCoeff

The Whitening coefficient, value range: 0x0~0x7f. If the value is bigger than 0x7f, the whitening is off. This parameter is effective in the test action: "Packet Tx", "Continue Tx" and "Packet Rx". For BT 4.0 LE test in "BT DUT Test Mode", don't care this parameter and should be set to 0xff.

BDAddress[12]

Value range: 0x000000000000~0xfffffffffff. The BD Address is used for generating the access code portion of the Tx packet for test actions: "Packet Tx" and "Continue Tx", and be an access code hit target for "Packet Rx" mode.

Return Values

If the operation completes successfully, the return value is TRUE. If the operation fails, the return value is FALSE and call [RTKBT_GetLastError\(\)](#) to retrieve the error code.

Remarks

This function should be called to setup the Tx/Rx parameters before run the [RTKBT_StartPacketTx\(\)](#), [RTKBT_StartContinueTx\(\)](#) and [RTKBT_StartRx\(\)](#). To avoid calling this function after the test action is running.

Example

```
//This example shows how to setup a Tx parameters and run a "Packet Tx" test until meet the
// target power.
```

```
RTKBT_TRXPARAM params;
```

```
params.Rate = 1;           //Rate = 1M
params.Channel = 40;       //channel = 40
params.TxPktCnt = 3000;    //Tx packet count = 3000
params.TxPktInterval = 2;  //Tx packet interval = (2+1) * 100us
params.PayloadType = 2;    //Payload Type = 0101
params.PayloadLength = 2500; //Patload length = 2500 bits
params.PacketHeader = 0x3ffff; //Packet Header = 0x3ffff
params.WhitenCoeff = 0x7f;  //Whiten on and Coeff = 0x7f
TxIndex = 5;
```

```
strcpy(params.BDAddress, "0000009e8b33"); //Set this for Access code
```

```
if (!RTKBT_SetTRxParameters(params))
{
```

```
printf("Set TRx Params Error Code = %d", RTKBT_GetLastError());
return;
}
else
{

    While(1)
    {
        if (!RTKBT_StartPacketTx())
            printf ("Start Packet Tx fail !!");
        else
        {
            //Place Get Tx Power Code here
            Tx_Power = GetTxPowerFromTester();

            //if Tx Power smaller than the target power (8 dBm)
            if ( (TxPower < 8) && (TxIndex <= 9) )
                RTKBT_TxPowerAdjust(++TxIndex);
            else //meet target power
                break;
        }
    }
}
```

1.8. RTKBT_StartPacketTx

bool RTKBT_StartPacketTx();

Parameters

None

Return Values

If the operation completes successfully, the return value is TRUE. If the operation fails, the return value is FALSE and the application can call [RTKBT_GetLastError\(\)](#) to retrieve the error code.

Remarks

This function initiates "Packet Tx" operation mode and can be used for RF Transmitter testing like Tx DEVM. Before starting "Packet Tx", the relative Tx parameters must be set by calling [RTKBT_SetTRxParameters\(\)](#). If there is any other test action still running, the application must call [RTKBT_StopTest\(\)](#) to stop it before calling this function to void redundancy action error.

Example

See the example at [RTKBT_SetTRxParameters\(\)](#).

1.9. RTKBT_StartContinueTx

bool RTKBT_StartContinueTx();

Parameters

None

Return Values

If the operation completes successfully, the return value is TRUE. If the operation fails, the return value is FALSE and the application can call [RTKBT_GetLastError\(\)](#) to retrieve the error code.

Remarks

This function initiates "Continue Tx" operation mode and can be used for RF Tx Power measurement. Before start "Continue Tx", the relative Tx parameters must be set by calling [RKBT_SetTRxParameters\(\)](#). If there is any other test action running, the application must call [RTKBT_StopTest\(\)](#) to stop it before calling this function to void redundancy test action error.

The "Continue Tx" mode could generate not only standard Bluetooth packets but also single carrier tone (setup follow 1. Rate = 1M 2. payload Type to "all 0's" (or "all 1's") and 3. Whitening off in [RTKBT_SetTRxParameters\(\)](#).

Example

//This example shows how to generate a single carrier tone at 2442MHz.

RTKBT_TRXPARAM params;

```
params.Rate = 1;           //Rate = 1M
params.Channel = 40;       //channel = 40
params.TxPktCnt = 0;       //Tx packet count, not necessary.
params.TxPKtInterval = 2;  //Tx packet interval, not necessary
params.PayloadType = 0;    //Payload Type = all 0's
params.PayloadLength = 2500; //Patload length = 2500 bits, not necessary
params.PacketHeader = 0x3ffff; //Packet Header = 0x3ffff
params.WhitenCoeff = 0xff;  //Whiten off
strcpy(params.BDAddress, "0000009e8b33"); //for access code

if (!RTKBT_SetTRxParameters(params)){
    printf("Set TRx Params Error Code = %d", RTKBT_GetLastError());
    return;
}else
{
    if (!RTKBT_StartContinueTx())
        printf ("Start Continue Tx fail !!");
    else
```



```
printf ("Start Continue Tx ok !!");  
}
```

REALTEK Confidential

1.10. RTKBT_StartRx

bool RTKBT_StartRx();

Parameters

None

Return Values

If the operation completes successfully, the return value is TRUE. If the operation fail, the return value is FALSE and the application can call [RTKBT_GetLastError\(\)](#) to retrieve the error code.

Remarks

This function initiates "Packet Rx" test action and can used for Rx BER testing. Before start "Packet Rx", the relative Rx parameters must be set for BER calculation by calling [RTKBT_SetTRxParameters\(\)](#). If there is any other test action running, the application must call [RTKBT_HCIReset\(\)](#) or [RTKBT_StopTest\(\)](#) to stop it before calling this function to void redundancy test action error.

Example

//This example shows how to receive 1M packets at 2442MHz

RTKBT_TRXPARAM params;

```
params.Rate = 1;           //Rate = 1M
params.Channel = 40;       //channel = 40
params.TxPktCnt = 0;       //Tx packet count, not necessary.
params.TxPKtInterval = 0;  //Tx packet interval, not necessary
params.PayloadType = 0;    //Payload Type = all 0's
params.PayloadLength = 2500; //Patload length = 2500 bits
params.PacketHeader = 0x3ffff; //Packet Header = 0x3ffff
params.WhitenCoeff = 0xff;  //Whiten off
strcpy(params.BDAddress, "0000009e8b33"); //for access code hit target
```

```
if (!RTKBT_SetTRxParameters(params)){
    printf("Set TRx Params Error Code = %d", RTKBT_GetLastError());
    return;
}else
{
    if (!RTKBT_StartRx())
        printf ("Start Packet Rx fail !!");
    else
        printf ("Start Packet Rx ok !!");
}
```

1.11. RTKBT_StopTest

bool RTKBT_StopTest();

Parameters

None

Return Values

If the operation completes successfully, the return value is TRUE. If the operation fail, the return value is FALSE and the application can call [RTKBT_GetLastError\(\)](#) to retrieve the error code.

Remarks

This function terminates test action for "Packet Tx", "Continue Tx" and "Packet Rx" mode. It also stops the statistics counter. To avoid getting the incomplete test result, the application should call [RTKBT_QueryTRxStatus\(\)](#) to determine that if the test is finished.

Example

//This example shows how to start "Packet Rx" test, get statistics, query running status and stop test.

```
RTKBT_TRXSTATISTICS statistics;
```

```
if (!RTKBT_StartPacketRx()) {  
    printf ("Start Packet Rx fail !!");  
    return;  
}
```

```
else{
```

```
    RTKBT_ClearCounter();  
    while(RTKBT_QueryTRxStatus()) {
```

```
        //place get statistics or BT tester control code here!!
```

```
        RTKBT_GetStatistics(&statistics);
```

```
        printf("Rx bits      = %f \\  
              Error bits   = %f \\  
              RxBER        = %f %% \\  
              RSSI         = %f dBm \\  
              CFO          = %f KHz \\  
              SignalQuality = %f \\  
              Tx bits      = %f ", statistics.RxBits, statistics.ErrorBits, statistics.RxBER,  
                                statistics.RSSI, statistics.CFO, statistics.SignalQuality,  
                                statistics.TxBits);
```

```
        If (UserStop)  
            break;
```

```
    }
```

```
    RTKBT_StopTest();  
}
```

REALTEK Confidential

1.12. RTKBT_QueryTRxStatus

bool RTKBT_QueryTRxStatus();

Parameters

None

Return Values

If the DUT stays in waiting HCI event state or Tx/Rx running state, the return value is TRUE. Otherwise, the return value is FALSE and the application can call [RTKBT_GetLastError\(\)](#) to retrieve the error code.

Remarks

For "Bluetooth DUT Test Mode", this function can be used to distinguish between Standby mode and non-Standby mode that defined in the Bluetooth standard,

For non-link mode test, this function can be used to determine whether the test is finished.

Example

See the example in [RTKBT_StopTest\(\)](#).

1.13. RTKBT_GetStatistics

bool RTKBT_GetStatistics(RTKBT_TRXSTATISTICS* statistics);

Parameters

Statistics is a RTKBT_TRXSTATISTICS structure defined in mp8723xBT.h, and it is used to return the test data. The members are all "double" type and listed as below:

RxBits → Total received bits.

Errorbits → Total received error bits.

RxBER → Received bit error rate in %.

RSSI → Receive signal strength in dBm

CFO → Received center frequency offset in KHz.

SignalQuality → Received signal quality index.

TxBits → Total transmitted bits. (always return 0)

Return Values

If the operation completes successfully, the return value is TRUE. If the operation fails, the return value is FALSE and the application can call [RTKBT_GetLastError\(\)](#) to retrieve the error code.

Remarks

This function is only effective for the non-link mode test. These counters can be reset by calling [RTKBT_ClearCounter\(\)](#).

Example

See the example in [RTKBT_StopTest\(\)](#).

1.14. RTKBT_ClearCounter

bool RTKBT_ClearCounter();

Parameters

None

Return Values

If the operation completes successfully, the return value is TRUE. If the operation fails, the return value is FALSE and the application can call [RTKBT_GetLastError\(\)](#) to retrieve the error code.

Remarks

This function resets the statistics counter that produced by the test actions: "Packet Tx" and "Packet Rx".

Example

See the example in [RTKBT_StopTest\(\)](#).

1.15. RTKBT_TxPowerAdjust

bool RTKBT_TxPowerAdjust(UINT8 FineTuneIndex);

Parameters

FineTuneIndex

For 8723a:

The Tx Power fine tune index, value range: 1~9.

For other RTK BT chip:

The Tx Power fine tune index, value range: 1~5.

Return Values

If the operation completes successfully, the return value is TRUE. If the operation fails, the return value is FALSE and the application can call [RTKBT_GetLastError\(\)](#) to retrieve the error code.

Remarks

This function is used to fine tune the RF Tx output Power.

Example

See the example at [RTKBT_SetTRxParameters\(\)](#)

1.16. RTKBT_ReadThermalMeter

UINT8 RTKBT_ReadThermalMeter();

Parameters

None

Return Values

If the operation completes successfully, the function returns the thermal meter value. If the operation fails, the return value is zero and the application can call [RTKBT_GetLastError\(\)](#) to retrieve the error code.

Remarks

This function can be used to get the reading value of the BT built-in thermal meter. If there is any other test action running, the application must call [RTKBT_HCIReset\(\)](#) or [RTKBT_StopTest\(\)](#) to stop it before calling this function to void redundancy test action error.

Example

```
int val;

if (RTKBT_QueryTRxStatus())
    RTKBT_StopTest();           //Stop the current test

val = RTKBT_ReadThermalMeter()

if (val == 0)
    printf("Read Thermal Error Code = %d", RTKBT_GetLastError());
else
    printf("Thermal value = %d", val);
```


1.17. RTKBT_WriteReg

*bool RTKBT_WriteReg(UINT32 Type, UINT32 RegAddr,
UINT32 RegValue, UINT32 Bitmask);*

Parameters

Type

Value range: 0~2. The register type: 0→RF, 1→Modem, 2→Global.

RegAddr

Value range: RF→ 0x0~0x3f, Modem→0x0~0x7f, Global:0x0~0x3ff. The register address.

RegValue

Value range: RF, Modem→ 0x0~0xffff, Global→ 0x0~0xffffffff. The register value.

BitMask

Value range: RF, Modem→0x0~0xffff, Global→0x0~0xffffffff. The bit position to write. The bit position must be consecutive. For example: 0x7c00 is valid, 0x4f00 is not valid.

Return Values

If the operation completes successfully, the function returns TRUE. If the operation fails, the return value is FALSE and the application can call [RTKBT_GetLastError\(\)](#) to retrieve the error code.

Remarks

This function can be used to write the RF/Modem/Global registers for internal debug purpose. To avoid unexpected error, it is recommendable to access RF registers without test action running.

Example

```
// This example shows how to write 4b'0110 to Modem register 0x2[12:9].
```

```
if (!RTKBT_WriteReg(1,0x2,0x6,0x1e00))  
    printf("Write Modem Register Error Code = %d", RTKBT_GetLastError());  
else  
    printf("Write Modem Register ok !!");
```

1.18. RTKBT_ReadReg

***UINT32 RTKBT_ReadReg(UINT32 Type, UINT32 RegAddr,
UINT32 Bitmask);***

Parameters

Type

Value range: 0~2. The register type: 0→RF, 1→Modem, 2→Global.

RegAddr

Value range: RF→ 0x0~0x3f, Modem→0x0~0x7f, Global:0x0~0x3ff. The register address.

BitMask

Value range: RF, Modem→0x0~0xffff, Global→0x0~0xffffffff. The bit position to read. The bit position must be consecutive. For example: 0x7c00 is valid, 0x4f00 is not valid.

Return Values

If the operation completes successfully, the function returns the register value. If the operation fails, the return value is zero and the application can call [RTKBT_GetLastError\(\)](#) to retrieve the error code.

Remarks

This function can be used to read the RF/Modem/Global registers for internal debug purpose. To avoid unexpected error, it is recommendable to access RF registers without test action running.

Example

```
// This example shows how to read 6b'100110 to RF register 0x3c[15:10].
```

```
if (!RTKBT_WriteReg(0,0x3c,0x26,0xfc00))  
    printf("Write RF Register Error Code = %d", RTKBT_GetLastError());  
else  
    printf("Write RF Register ok !!");
```

1.19. RTKBT_SendHCICmd

***UINT8* RTKBT_SendHCICmd(UINT16 nOpcode, UINT8 nLength,
UINT8* lpParam, UINT8 nEventcode);***

Parameters

nOpcode

Value range: see the Opcode definition in the Bluetooth specification. The 2 byte Opcode used to uniquely identify different types of HCI commands. It is composed of the 6 bits Opcode Group Field (OGF) and 10 bits OpCode Command Field (OCF).

nLength

The parameter length of the HCI command, value range: 0~255. 0 means no parameter required.

lpParam

The parameter content is constructed of an 8-bit array. The array size must equal to the value that specified in the nLength field.

nEventcode

Value range: see the Event code definition in the Bluetooth specification. It specifies the desired HCI Event after sending HCI command.

Return Values

If the operation completes successfully, the function returns the HCI Event parameters in an 8-bit array that specified in the Event code fields. If the operation fails, the return value is an all zero array and the application can call [RTKBT_GetLastError\(\)](#) to retrieve the error code. The length of the return array must be equal to 257 bytes exactly to avoid memory operation error.

Remarks

This function is used to send any HCI command that is supported by Bluetooth specification.

This function is only support for HostType = 0~2 (specified in [RTKBT_OpenAdapter\(\)](#)).

Example

```
// This example shows how to get the BD address of the DUT.
```

```
UINT8 ReturnEvent[257];  
UINT8 Param[255], status;  
UINT16 OPCode;
```

```
memset(ReturnEvent, 0, 257*sizeof(UINT8)); //set the return array to zero  
memset(Param, 0, 255*sizeof(UINT8));      //set the parameter array to zero
```

```
//no parameter required for this HCI command, and wait 0xe event (Command complete Event)
ReturnEvent = RTKBT_SendHCICmd(0x1009,0, Param, 0xe);
```

```
status = ReturnEvent [5];
```

```
OPCode = (ReturnEvent [4] << 8) + ReturnEvent [3];
```

```
if ( (status == 0x0) && (OPCode == 0x1009) )
```

```
{
```

```
    sprintf("Bluetooth Device Address: 0x%02X%02X%02X%02X%02X%02X",
```

```
            *(ReturnEvent +11),*(ReturnEvent +10),*(EventBuf+9),
```

```
            *(ReturnEvent +8),*(ReturnEvent +7),*(ReturnEvent +6));
```

```
}
```

```
else
```

```
    sprintf("Unknown Bluetooth Device Address !!");
```

2. Appendix

2.1.Error Code Descriptions

Value/Constant	Description
0 (0x0) --- eNoError	The operation completed successfully.
1 (0x1) --- eAdapterInitialFail	Adapter initial fail.
2 (0x2) --- eInvalidHostType	Incorrect host interface type
3 (0x3) --- eInvalidUSBPortNumber	Incorrect USB port number. Valid value: 1~2
4 (0x4) --- eInvalidUARTPortNumber	Incorrect UART port number. Valid value: 1~12
5 (0x5) --- eInvalidPClePortNumber	Incorrect PCle port number. Valid value: 1~2
6 (0x6) --- eInvalidUARTBaudRate	Incorrect UART baud rate. The valid rates are 9600, 14400, 19200, 38400, 57600, 115200
7 (0x7) --- eNotRTKBTChip	This is not Realtek BT controller, thus some functions can not support.
8 (0x8) --- eInvalidDataRate	Incorrect Data Rate. Valid value: 1~3
9 (0x9) --- eInvalidTxPktInterval	Incorrect Tx Packet Interval. Valid value: 0~15
10 (0xa) --- eInvalidChannelNumber	Incorrect Channel Number. Valid value: 0~78
11 (0xb) --- eInvalidPayloadType	Incorrect Payload Type. Valid value: 0~4
12 (0xc) --- eInvalidPayloadLength	Incorrect Payload Length. Valid value: 1~8191
13 (0xd) --- eInvalidPacketHeader	Incorrect Packet Header. Valid value: 0~0x3ffff
14 (0xe) --- eInvalidBDAddress	Incorrect BD Address. Valid value: 0x000000000000~0xffffffffffff
15 (0xf) --- eInvalidRegisterType	Incorrect Register Type. Valid value: 0~2

16 (0x10) --- eInvalidTxIndex	Incorrect Tx Index. Corse Tx Index value: 0~19, Fine Tx Index Value: 1~9
17 (0x11) --- eHCIResetFail	HCI Reset command fail.
18 (0x12) --- eEnableDUTModeError,	HCI DUT Test mode command fail.
19 (0x13) --- eRedundancyTestAction	There is another excuting test action
20 (0x14) --- ePacketTxFail	Packet Tx fail
21 (0x15) --- eContinueTxFail	Continue Tx fail
22 (0x16) --- ePacketRxFail	Packet Rx fail
23 (0x17) --- eStopFail	Stop test fail
24 (0x18) --- eCFOAdjustFail	Adjust crystal capacitor fail
25 (0x19) --- eNotSupport	Not support in this test mode.
26 (0x1a) --- eUnknowError	Unspecified error
27 (0x1b) --- eNoHCIEEvtResponse	No BT HCI Event return
28 (0x1c) --- eHostInitFail	Host initial fail
29 (0x1d) --- eInvalidTxGainIndex	Invalid Tx gain index
30 (0x1e) --- eH5toH4TranslateFail	H5-H4 translate fail
31 (0x1f) --- eTestActionIsStillRunning	Test action is still running

2.2. The DLL File

The essential Bluetooth MP DLL files are listed as below:

1. mp8723xBT.dll: The main DLL file.
2. mp8723xBT.lib: The library file
3. mp8723xBT.h: The header file, also define some structures.
4. rtkbt.dll: The DLL file for translating MP dll to bus driver.
5. \RTKBT*.txt, *.ini: The ini files required for RTK Test Mode.

2.3.Example

2.3.1. BT DUT Test Mode (Link Mode Test), Host Interface: BT-USB

```
#include "mp8723xBT.h"

//open Port1 USB as Host interface
if (!RTKBT_OpenAdapter(0,1,0))
    printf("Open Adapter Error = %d", RTKBT_GetLastError());
else
    printf("Initial Adapter ok !!");

if (!RTKBT_BTTestMode(0))
{
    printf("BT Test Mode Error = %d", RTKBT_GetLastError());
    return;
}
else
{
    printf("BT DUT Test Mode ok !!");
}

//Place BT tester control code here

//Exit if test finish
//Issue an HCI Reset
if (!RTKBT_HCIReset())
    printf("HCI Reset Error = %d", RTKBT_GetLastError());
else
    printf("HCI Reset ok !!");

//Close Port1 USB as HCI interface
if (!RTKBT_CloseAdapter())
    printf("Close Adapter Error = %d", RTKBT_GetLastError());
else
    printf("Close Adapter ok !!");
```


2.3.2. Packet-Tx Test (Non-Link Mode Test), Host Interface: WiFi-PCIe

(For Tx Modulation Characteristics test Δf_1 , 2442MHz/DH5/Whiten-off/11110000/2712bits)

```
#include "mp8723xBT.h"

RTKBT_TRXPARAM params;

//open WiFi-PCIe as Host interface
if (!RTKBT_OpenAdapter(6,0,0))
    printf("Open Adapter Error = %d", RTKBT_GetLastError());
else
    printf("Initial Adapter ok !!");

//Setup Tx packet parameters
params.Rate = 1; //Rate = 1M
params.Channel = 40; //channel = 40
params.TxPktCnt = 0; //Tx packet count, 0→ continue packet Tx
params.TxPktInterval = 2; //Tx packet interval, not necessary
params.PayloadType = 3; //Tx Payload Type = 11110000
params.PayloadLength = 2712; //Payload length = 2712 bits, not necessary
params.PacketHeader = 0x3ffff; //Packet Header = 0x3ffff
params.WhitenCoeff = 0xff; //Whiten off
strcpy(params.BDAddress, "0000009e8b33"); //for access code
params.TxGainIndex = 7; //Tx Gain Index Range: 0~7

if (!RTKBT_SetTRxParameters(params))
{
    printf("Set TRx Params Error Code = %d", RTKBT_GetLastError());
    return;
}

if (!RTKBT_StartPacketTx())
{
    printf("Start Packet Tx Error = %d", RTKBT_GetLastError());
    return;
}
else
{
    printf("BT Start Packet Tx ok !!");
}

If (!RTKBT_QueryTRxStatus()) //check if packet Tx is running
    return;

//Place BT tester control code here
```

```
//Exit if test finish
if (!RTKBT_StopTest())
    printf("Stop Test Error = %d", RTKBT_GetLastError());
else
    printf("Stop Test ok !!");

//Close Host interface
if (!RTKBT_CloseAdapter())
    printf("Close Adapter Error = %d", RTKBT_GetLastError());
else
    printf("Close Adapter ok !!");
```

REALTEK Confidential

2.3.3. Packet-Rx Test (Non-Link Mode Test), Host Interface: Android ADB

(For Receiving 2402MHz/3DH5/Whiten-off/1010/8168 bits/BD Address = 0x000000c6967e)

```
#include "mp8723xBT.h"

RTKBT_TRXSTATISTICS statistics;
RTKBT_TRXPARAM params;

//open android adb as Host interface
if (!RTKBT_OpenAdapter(7,0,0))
    printf("Open Adapter Error = %d", RTKBT_GetLastError());
else
    printf("Initial Adapter ok !!");

//Setup Rx packet parameters
params.Rate = 2;           //Rate = 3M
params.Channel = 0;        //channel = 0
params.TxPktCnt = 0;        //for Rx test, don't care it.
params.TxPKtInterval = 2;  //for Rx test, don't care it.
params.PayloadType = 3;     //Payload Type = 1010
params.PayloadLength = 8168; //Payload length = 8168 bits, not necessary
params.PacketHeader = 0x3ffff; //Packet Header = 0x3ffff
params.WhitenCoeff = 0xff;   //Whiten off
strcpy(params.BDAddress, "000000c6967e"); //for access code
params.TxGainIndex = 7;     //for Rx test, don't care it

if (!RTKBT_SetTRxParameters(params))
{
    printf("Set TRx Params Error Code = %d", RTKBT_GetLastError());
    return;
}

if (!RTKBT_StartPacketRx())
{
    printf("Start Packet Rx Error = %d", RTKBT_GetLastError());
    return;
}
else
{
    printf("BT Start Packet Rx ok !!");
}

If (!RTKBT_QueryTRxStatus()) //check if packet Rx is running
    return;
```

```
RTKBT_ClearCounter();

//place get statistics or BT tester control code here!!

While (RTKBT_QueryTRxStatus()) {

    RTKBT_GetStatistics(&statistics);
    printf("Rx bits      = %f \
          Error bits   = %f \
          RxBER        = %f %% \
          RSSI         = %f dBm \
          CFO          = %f KHz \
          SignalQuality = %f \
          Tx bits      = %f ", statistics.RxBits, statistics.ErrorBits, statistics.RxBER,
          statistics.RSSI, statistics.CFO, statistics.SignalQuality,
          statistics.TxBits);

    If (UserStop)
        break;
}

//Exit if test finish
if (!RTKBT_StopTest())
    printf("Stop Test Error = %d", RTKBT_GetLastError());
else
    printf("Stop Test ok !!");

//Close Host interface
if (!RTKBT_CloseAdapter())
    printf("Close Adapter Error = %d", RTKBT_GetLastError());
else
    printf("Close Adapter ok !!");
```