

BQB bluetooth Test mode User Guide

Module port description

The DUT module should connect to two uart for testing, one is UART0 and the other is UART1.

UART0

The default serial port for downloading bin files, printing, and receiving commands. The baud rate is 115200. The pins used are:

- TXD0 : connected to the USB serial port RXD,
- RXD0 : connected to the USB serial port TXD.

UART1

HCI serial port for connecting to HCI host. When classic BT is tested, UART1 is connected to the PC, and HCI host is simulated using the python script tinyBH. When measuring BLE, UART1 is connected to the tester (Also you can use this tool to make DUT into BLE direct test mode). UART1 requires hardware flow-control.

The baud rate is 115200, hardware flow control is enabled. The pins for UART1 can be configured through commands via UART0. The default pins are:

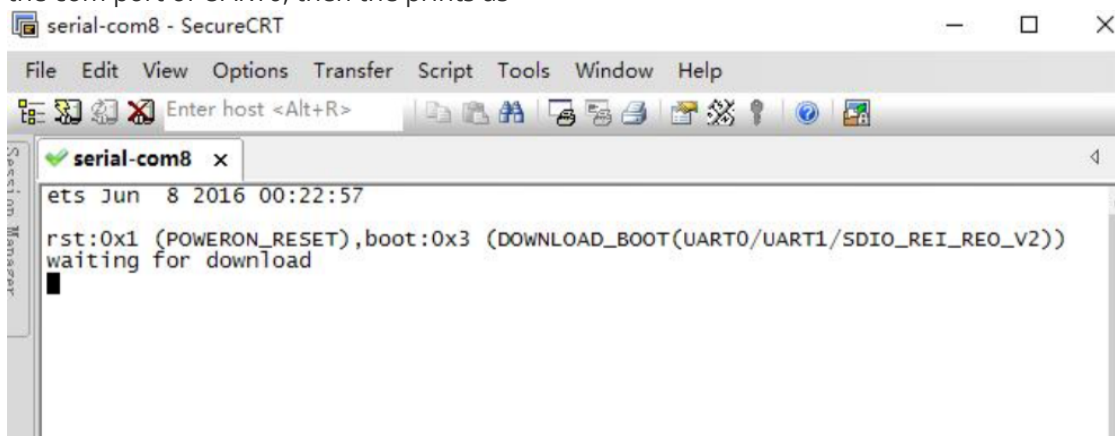
- pin IO5 : TXD of DUT, connected to the USB serial port RXD
- pin IO18 : RXD of DUT, connected to the USB serial port TXD
- pin IO19 : RTS of DUT, connected to the USB serial port CTS
- pin IO23 : CTS of DUT, connected to the USB serial port RTS

Pin usage

- pin IO0 : Start mode selection. Connect to GND and the chip starts in download mode. Connected to 3.3V or left floating, the chip starts in flash mode (working mode).

Device Boot

1. Download the bin file
2. Module IO0 is connected to 0, UART0 is connected to the computer, and the module is powered on. At this point the chip enters the download mode, use a UART tool to connect the com port of UART0, then the prints as

A screenshot of a SecureCRT terminal window titled 'serial-com8 - SecureCRT'. The window has a menu bar with 'File', 'Edit', 'View', 'Options', 'Transfer', 'Script', 'Tools', 'Window', and 'Help'. Below the menu bar is a toolbar with various icons. The terminal text shows the date and time 'ets Jun 8 2016 00:22:57', followed by a reset message 'rst:0x1 (POWERON_RESET),boot:0x3 (DOWNLOAD_BOOT(UART0/UART1/SDIO_REI_REO_V2))', and then 'waiting for download' with a cursor. On the left side of the terminal window, there is a vertical label 'Serial Monitor'.

3. Open flash_download_tools_v3.6.8.exe

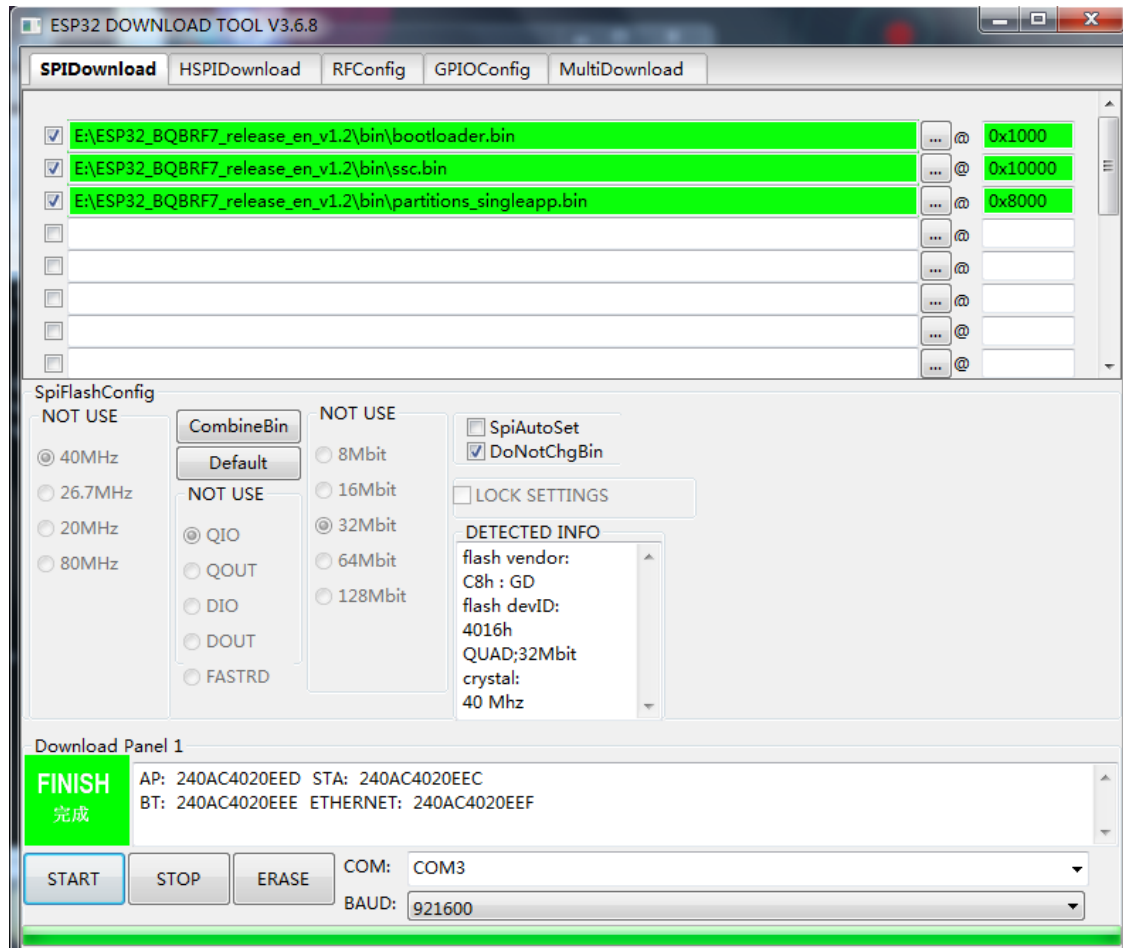
4. Select ESP32 DownloadTool

There are three bin files in directory bin:

- Bootloader.bin
- partitions_singleapp.bin
- ssc.bin

Download these bin files as your own absolute path instead of the path

'E:\ESP32_BQBRF7_release_en_v1.2\bin' shown in the picture via the com port of UART0.



Note: The checkbox "DoNotChgBin" needs to be selected.

5. Click start to download. After the download is complete, power off, let IO0 float, then power on, the chip will enter flash boot mode (working mode).

Run the tool

1. IO0 is left floating, UART0 is connected to PC, and the RF signal line is connected.
2. Set up UART1

Normally, if you test classic BT, UART1 should be connected to the PC. If you want to test BLE, UART1 should be connected to the tester. Then power on.

BTW, If you just want to test DUT RX/TX without tester, you can also use this tool to make DUT enter into BLE Direct test mode.

For this module, you need to assign the pins for UART1 yourself and do not use the default pins(i.e. IO5, IO18, IO19, IO23). You need to configure the pins used via UART0 commands. For example, if you want to use IO21, IO22, IO19 and IO5 the UART1 TXD, RXD, RTS, CTS pins, you need to:

- connect IO21 to the USB serial port RXD

- connect IO22 to the USB serial port TXD
- connect IO19 to the USB serial port CTS
- connect IO5 to the USB serial port RTS
- After power up, type the following command via UART0:
 - `bqb -z set_uart_pin -t 21 -r 22 -q 19 -c 5`

3. Software configuration

To configure the test software, you need to type the following commands via UART0:

- `bqb -z set_ble_tx_power -i 4`
 - `bqb -z set_power_class -i 3 -a 4`
 - `bqb -z set_pll_track -e 0`
 - `bqb -z init`
4. If you test BLE, skip step #5 and #6. Normally, if test BLE, UART1 will connect to tester mentioned in step #2. Except that you want to use this tool to test DUT RX/TX performance without BQB tester.
 5. Modify the `*/tools/HCI_host/dev0.conf`: modify the the value of `UART_PORT` to the COM port of UART1.
 6. If test BR/EDR, Open `*/tools/HCI_host/tinyBH.exe` and then press the command(`hci dut` is very important, if not set name, it may crash):
 - `hci reset`
 - `hci set_evt_mask`
 - `hci set_name ESPTEST`
 - `hci dut`
 - `hci ipscan`
 7. At this point the chip has entered test mode. You can start testing.

Some Known issues

TP/RCV/CA/BV-04-C [Blocking Performance] Precautions

When running the blocking test, when sweeping between 3280MHz and 3285MHz, the firmware will have a high probability of exception. This is the need to power cycle and repeat 4. Start and run. The test can still proceed.

Command Description of UART0

1. Set BLE TX power

Command : `bqb -z set_ble_tx_power -i [Power_level_index]`

Param:

- **Power_level_index** : Range [0 ~ 7].
 - 0 : -12dbm
 - 1 : -9dbm
 - 2 : -6dbm
 - 3 : -3dbm
 - 4 : 0dbm
 - 5 : +3dbm
 - 6 : +6dbm

- 7 : +9dbm

Description : Set BLE Tx power by power level index.

Usage: bqb -z set_ble_tx_power -i 4

Set BLE TX power to 0dbm

2. Set Classic Power Class

Command : bqb -z set_power_class -i [Min_power_level_index] -a [Max_power_level_index]

Param:

- **Min_power_level_index** : Range [0 ~ 7].
 - 0 : -12dbm
 - 1 : -9dbm
 - 2 : -6dbm
 - 3 : -3dbm
 - 4 : 0dbm
 - 5 : +3dbm
 - 6 : +6dbm
 - 7 : +9dbm
- **Max_power_level_index** : Range [0 ~ 7].
 - 0 : -12dbm
 - 1 : -9dbm
 - 2 : -6dbm
 - 3 : -3dbm
 - 4 : 0dbm
 - 5 : +3dbm
 - 6 : +6dbm
 - 7 : +9dbm

Description : Set Classic BT Tx power range by power level index. The min power level index should be less than or equal to max power level index.

Usage: bqb -z set_power_class -i 3 -a 4

Set Classic BT TX minimum power to -3dbm and maximum power to 0dbm

3. Disable PLL track function

Command : bqb -z set_pll_track -e [enable]

Param:

- **enable** : 0 or 1.
 - 0 : disable
 - 1 : enable

Description : Set BLE Tx power by power level index. In Test mode, this function must be disabled.

Usage: bqb -z set_pll_track -e 0

Disable PLL Track function.

4. Initialise bluetooth controller dual mode

Command : bqb -z init

Description : Initialise bluetooth controller dual mode.

5. Set UART1 PIN

Command : `bqb -z set_uart_pin -t [TX_pin] -r [RX_pin] -q [RTS_pin] -c [CTS_pin]`

Param:

- **TX_pin** : GPIO number as you want. The range according to the ESP32 datasheet.
- **RX_pin** : GPIO number as you want. The range according to the ESP32 datasheet.
- **RTS_pin** : GPIO number as you want. The range according to the ESP32 datasheet.
- **CTS_pin** : GPIO number as you want. The range according to the ESP32 datasheet.

Description : Set GPIO pin for UART1.

Usage: `bqb -z set_uart_pin -t 5 -r 18 -q 19 -c 23`

Set the GPIO5 as TX, GPIO18 as RX, GPIO19 as RTS and GPIO23 as CTS for UART1.

6. Set UART1 Param (Baudrate and HW Flow control)

Command : `bqb -z set_uart_param -f [enable] -b [baudrate]`

Param:

- **eanbe** : 0 or 1
 - 0 : disable
 - 1 : enable
- **baudrate** : Suggest use 115200 or 921600. Other value according to the ESP32 datasheet and your uart chip datasheet.

Description : Set UART1 parameter, include baudrate and Hardware flow control.

Usage: `bqb -z set_uart_param -f 0 -b 921600`

Set UART1 baudrate to 921600 and disable hardware flow control

Command Description of main.exe (HCI via UART1)

1. RESET Controller

Command : `hci reset`

Description : Reset all the bluetooth controller.

2. Set legacy event mask

Command : `hci set_evt_mask`

Description : Set legacy event mask for bluetooth controller. It will use the default mask value, so this command has no parameter. If doesn't send this command, some event may not be send up from controller

3. Set LE event mask

Command : `hci le_set_evt_mask`

Description : Set LE event mask for bluetooth controller. It will use the default mask value, so this command has no parameter. If doesn't send this command, some event may not be send up from controller. Such as advertising report.

4. Set device name

Command : `hci set_name [name]`

Param:

- **name** : Characters of name.

Description : Set device name. In Classic BT test mode, the device name must not be empty.

Usage: hci set_name ESPTEST

5. Make Classic BT enter DUT mode (Under test mode).

Command : hci dut

Description : Make Classic BT enter into "Under test" mode. When the DUT in this mode, then the tester could send test command via LMP.

6. Make Classic BT enter inquiry scan and page scan state.

Command : hci ipscan

Description : When DUT in inquiry scan and page scan state, the tester(bluetooth master) could connect to the DUT.

7. Enter Direct Test RX mode

Command : hci le_rx_test [channel(Mhz)]

Param:

- **channel(Mhz)** : Range[2402 ~ 2480], step 2Mhz

Description : Enter Direct Test RX mode with the channel(Mhz).

Usage: hci le_rx_test 2480

```
* RX packets in channel 2480Mhz
```

8. Enter Direct Test TX mode

Command : hci le_tx_test [channel(Mhz)] [test_data_length] [payload_type]

Param:

- **channel(Mhz)** : Range[2402 ~ 2480], step 2Mhz
- **test_data_length** : Range[0 ~ 255].
- **payload_type** : Enumeration value following.
 - **0x00** : PRBS9 sequence '11111111100000111101...
 - **0x01** : Repeated '11110000' (in transmission order)
 - **0x02** : Repeated '10101010' (in transmission order)
 - **0x03** : PRBS15 sequence as described
 - **0x04** : Repeated '11111111' (in transmission order) sequence
 - **0x05** : Repeated '00000000' (in transmission order) sequence
 - **0x06** : Repeated '00001111' (in transmission order) sequence
 - **0x07** : Repeated '01010101' (in transmission order) sequence

Description : Enter Direct Test TX mode with the channel(Mhz).

Usage: hci le_tx_test 2480 10 0x4

```
* TX packets in channel 2480Mhz with data length 10bytes of data type 0x04
```

9. Exist Direct Test mode

Command : hci le_test_end

Description : Exit Direct Test mode, it will report test result in console if exit from RX test mode.

Sample Log:

```
[NORMAL][I][    CFG] : ===== Global Config Dump [START] =====
[NORMAL][I][    CFG] : Device Name : dev0
[NORMAL][I][    CFG] : Mode : HciConsole
[NORMAL][I][    CFG] : Layer: HciOnly
[NORMAL][I][    CFG] : ===== Global Config Dump [END] =====
[NORMAL][I][  DEVICE] : Device initialising [dev0] ...
[MISC]
[NORMAL][I][    EIF] : EIF init
[ dev0][I][  HUART] : Open uart
[NORMAL][I][ VTHREAD] : vthread[dev0 [HCI]] running!!
[NORMAL][I][ VTHREAD] : vthread[dev0 [HCI]] running!!
[ dev0][I][ GAP_BLE] : module "gap" \init
[NORMAL][I][ VTHREAD] : vthread[dev0[STACK]] running!!
[NORMAL][I][ VTHREAD] : vthread[dev0[SYS_EVT]] running!!
=====
[ Cns1][I][ CONSOLE] : Console mode starting ...
>>
>>
>>hci reset
[ Cns1][I][ CONSOLE] : console echo: hci reset
[ Cns1][I][HCI_CNSL] : hci_console_handler: reset
[ Cns1][I][HCI_CNSL] : send reset
[ dev0][D][    HCI] : 01 03 0c 00
>>[ dev0][D][    H4] : 01 03 0c 00
[ dev0][D][    H4] : [DEV : dev0] RECV:
[ dev0][D][    HCI] : HCI_EVT
[ dev0][D][    HCI] : EVT Opcode 0e
[ dev0][D][    HCI] : TOTAL LENGTH 4
[ dev0][D][    HCI] : HCI EVT COMMAND COMPLETE:
[ dev0][D][    HCI] : Num HCI COMMAND PACKETS 5
[ dev0][D][    HCI] : HCI EVT RESET, STATUS[00]
[ Cns1][I][HCI_CNSL] : SYS HCI EVT 10001

>>hci le_rx_test 2402
[ Cns1][I][ CONSOLE] : console echo: hci le_rx_test 2402
[ Cns1][I][HCI_CNSL] : hci_console_handler: le_rx_test
[ Cns1][I][HCI_CNSL] : le rx test
[ dev0][D][    HCI] : 1d 20 01 00
>> [ dev0][D][    H4] : [DEV : dev0] RECV:
[ dev0][D][    HCI] : HCI_EVT
[ dev0][D][    HCI] : EVT Opcode 0e
[ dev0][D][    HCI] : TOTAL LENGTH 4
[ dev0][D][    HCI] : HCI EVT COMMAND COMPLETE:
[ dev0][D][    HCI] : Num HCI COMMAND PACKETS 5
[ dev0][D][    HCI] : HCI EVT LE RX TEST, STATUS[00]
[ Cns1][I][HCI_CNSL] : SYS HCI EVT 2010010

>>
>>hci le_test_end
[ Cns1][I][ CONSOLE] : console echo: hci le_test_end
[ Cns1][I][HCI_CNSL] : hci_console_handler: le_test_end
```

```

[ Cns1][I][HCI_CNSL] : le test end
[ dev0][D][ HCI] : 1f 20 00
>> [ dev0][D][ H4] : [DEV : dev0] RECV:
[ dev0][D][ HCI] : HCI_EVT
[ dev0][D][ HCI] : EVT Opcode 0e
[ dev0][D][ HCI] : TOTAL LENGTH 6
[ dev0][D][ HCI] : HCI EVT COMMAND COMPLETE:
[ dev0][D][ HCI] : Num HCI COMMAND PACKETS 5
[ dev0][D][ HCI] : HCI EVT LE TEST END, STATUS[00], NUMBER_OF_PACKETS[0000]
[ Cns1][I][HCI_CNSL] : SYS HCI EVT 2010012

>>hci le_tx_test 2480 10 0x4
[ Cns1][I][CONSOLE] : console echo: hci le_tx_test 2480 10 0x4
[ Cns1][I][HCI_CNSL] : hci_console_handler: le_tx_test
[ Cns1][I][HCI_CNSL] : le tx test
39 10 4
[ dev0][D][ HCI] : 1e 20 03 27 0a 04
>> [ dev0][D][ H4] : [DEV : dev0] RECV:
[ dev0][D][ HCI] : HCI_EVT
[ dev0][D][ HCI] : EVT Opcode 0e
[ dev0][D][ HCI] : TOTAL LENGTH 4
[ dev0][D][ HCI] : HCI EVT COMMAND COMPLETE:
[ dev0][D][ HCI] : Num HCI COMMAND PACKETS 5
[ dev0][D][ HCI] : HCI EVT LE TX TEST, STATUS[00]
[ Cns1][I][HCI_CNSL] : SYS HCI EVT 2010011

>>
>>hci le_test_end
[ Cns1][I][CONSOLE] : console echo: hci le_test_end
[ Cns1][I][HCI_CNSL] : hci_console_handler: le_test_end
[ Cns1][I][HCI_CNSL] : le test end
[ dev0][D][ HCI] : 1f 20 00
>> [ dev0][D][ H4] : [DEV : dev0] RECV:
[ dev0][D][ HCI] : HCI_EVT
[ dev0][D][ HCI] : EVT Opcode 0e
[ dev0][D][ HCI] : TOTAL LENGTH 6
[ dev0][D][ HCI] : HCI EVT COMMAND COMPLETE:
[ dev0][D][ HCI] : Num HCI COMMAND PACKETS 5
[ dev0][D][ HCI] : HCI EVT LE TEST END, STATUS[00], NUMBER_OF_PACKETS[0000]
[ Cns1][I][HCI_CNSL] : SYS HCI EVT 2010012

```