RF Performance Test

User Manual

Version: 2.0

www.bouffalolab.com

Contents

1 Version record . . . 6
2 Overview . . . 7
3 Programming tool kit . . . 9
4 Program test firmware . . . 10
4.1 Program test firmware to BL702/704/706 IOT module (with Flash) . . . 10
4.2 Program test firmware to BL702L/704L IOT module (with Flash) . . . 13
5 Run test firmware . . . 17
6 Basic configuration . . . 18
7 Single Tone test . . . 23
8 802.15.4 test . . . 24
8.1 802.15.4 TX test . . . 24
8.2 802.15.4 RX test . . . 25
9 BLE test . . . 26
10 HBN/PDS test . . . 29
11 Programming guide for mass production test . . . 30
12 Serial communication commands . . . 33
12.1 Shakehand . . . 33
12.2 Get cap code . . . 33
12.3 Set cap code . . . 33
12.4 Get power . . . 33
12.5 Set power . . . 34
12.6 Set channel . . . 34
12.7 Set TX frequency . . . 34
12.8 Set sequence number . . . 34
12.9 TX . . . 34
12.10 RX . . . 34
12.11 HBN . . . 35
12.12 PDS . . . 35
12.13 Power Offset . . . 35
12.14 Single Tone . . . 35
12.15 CCA . . . 35
12.16 Active . . . 36
12.17 Read memory . . . 36
12.18 Write memory . . . 36

12.19 Write cap code to efuse register 36
12.20 Write power offset to efuse register (BL702) 36
12.21 Write power offset to efuse register (BL702L) 37
12.22 Write mac address to efuse register 37
12.23 Read cap code from efuse register 37
12.24 Read power offset from efuse register 37
12.25 Read mac address from efuse register 38
12.26 Program efuse 38
12.27 Cap code temperature calibration (BL702L) 38
12.28 Get MFG FW version 38
12.29 Get MFG FW build infomation 38
12.30 Get temperature 39
12.31 Get MAC address 39
12.32 Exit MFG FW 39
12.33 Reset chip 39
12.34 BLE Test 39

List of Figures

2.1 Interface of test tool 7

3.1 Programming tool kit 9

4.1 IOT Module Evaluation Kit 10
4.2 Adapter board 11
4.3 Programming interface 12
4.4 Interface of successful programming 13
4.5 IOT Module Evaluation Kit 14
4.6 Programming interface 15
4.7 Interface of successful programming 16

5.1 Log of successful operation 17

6.1 Get parameters 18
6.2 Set tx power 19
6.3 Set power offset 20
6.4 Update capacitance compensation value 21
6.5 Set active mode 22

7.1 Single Tone test 23

8.1 802.15.4 TX test 24
8.2 802.15.4 RX test 25

9.1 BLE Tx Payload Type 27
9.2 BLE Tx Rate 27
9.3 BLE Rx Rate 28
9.4 BLE Modulation Index 28

10.1 HBN/PDS parameter setting 29

11.1 Modify active entry 31
11.2 Programming interface for mass production test 32

1 Version record

Table 1.1: Version record

Version	Update content
V1.0	Initial release
V1.1	Add content related to power offset and efuse
V1.2	Add programming guide for mass production test
V1.3	Add read/write command for mac address in efuse
V1.4	Update pictures and 802.15.4 RX test chapter
V1.5	Add content related to BL702L
V1.6	Update commands related to BL702L
V2.0	Modify programming times of mac address in BL702L efuse

2 Overview

The RF performance test tool (RF MFG) is a tool provided by Bouffalo Lab for RF evaluation and testing. It includes two parts: a test tool and a test image (MFG Firmware). The interface of the test tool is shown in the figure below.

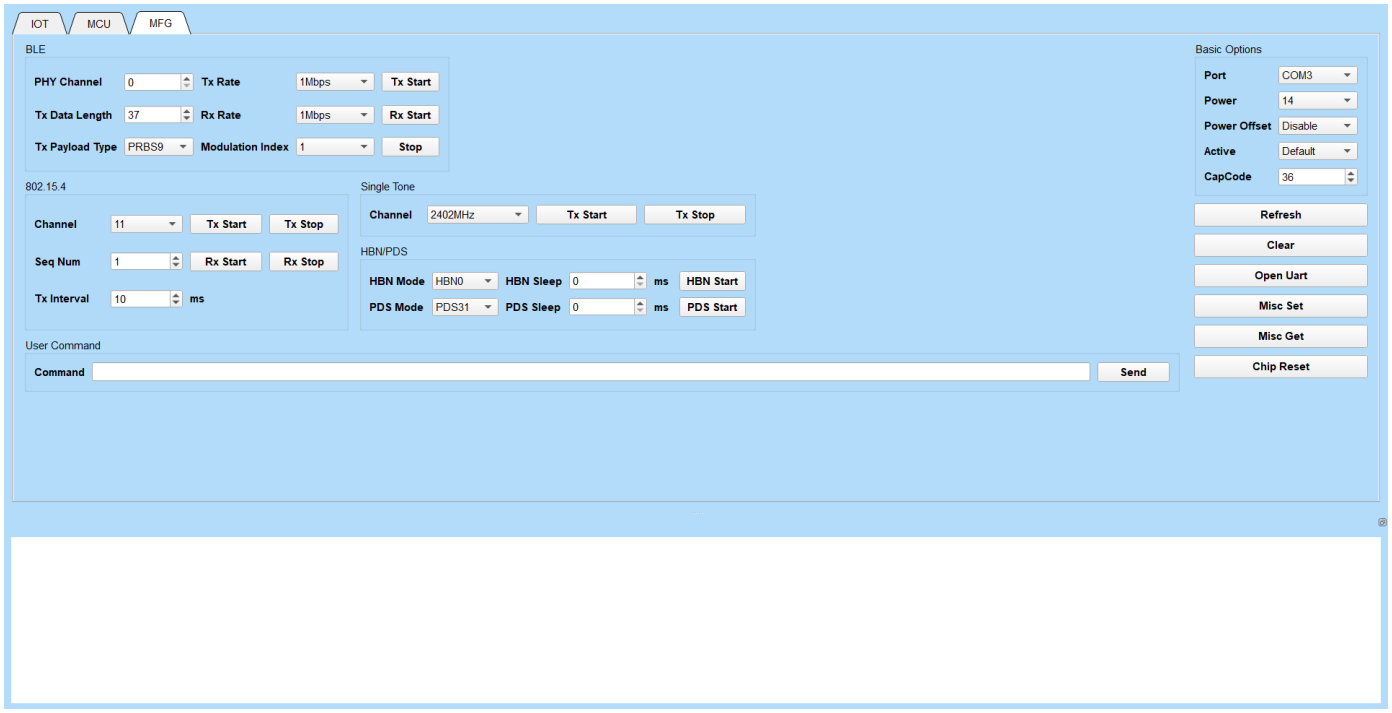

Fig. 2.1: Interface of test tool

The functions that can be performed by the RF performance test tool (RF MFG) include:

- Single Tone test
- 802.15.4 packet transmission
- 802.15.4 packet reception
- BLE packet transmission

- BLE packet reception
- PDS (Power Down Sleep) test
- HBN (Hibernate) test

3 Programming tool kit

If user does not have the programming tool kit (which is also called Dev Cube), please download it from Bouffalo Lab Dev Cube. It contains the RF test firmware, firmware programming tool, RF test tool, etc. The content of the tool kit is shown in the figure below. It is a tool set used by customers to develop various types of Bouffalo Lab chips.

名称	修改日期	类型	大小
chips	2022/12/1 14:26	文件夹	
docs	2022/12/1 14:27	文件夹	
utils	2022/12/1 14:27	文件夹	
bflb_iot_tool.exe	2022/12/1 14:10	应用程序	13,494 KB
bflb_iot_tool-macos	2022/12/1 14:20	文件	15,214 KB
bflb_iot_tool-ubuntu	2022/12/1 14:18	文件	15,490 KB
BLDevCube.exe	2022/12/1 14:10	应用程序	36,475 KB
BLDevCube-macos	2022/12/1 14:20	文件	42,602 KB
BLDevCube-macos-m1	2022/11/24 14:35	文件	29,973 KB
BLDevCube-ubuntu	2022/12/1 14:18	文件	62,751 KB
clear.bat	2022/11/17 16:19	Windows 批处理...	4 KB
ReleaseNote_bl.txt	2022/12/1 14:09	TXT 文件	21 KB

Fig. 3.1: Programming tool kit

4 Program test firmware

4.1 Program test firmware to BL702/704/706 IOT module (with Flash)

The following takes BL704 IOT module evaluation kit as an example to introduce the programming process.

The BL704 IOT module evaluation kit BL704_IoT_DevKit is shown in the figure below.

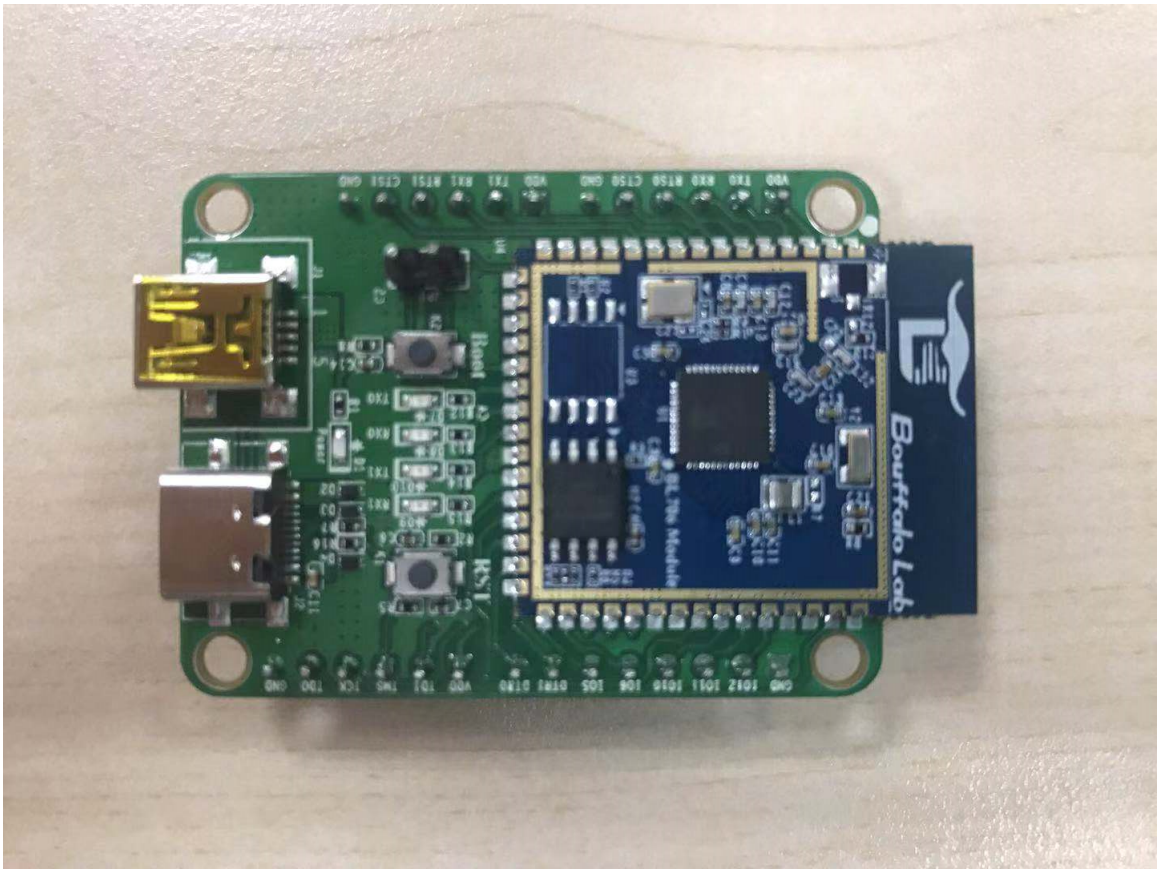

Fig. 4.1: IOT Module Evaluation Kit

The evaluation kit consists of an IOT module and a motherboard. The motherboard uses the USB interface for power

supply.

In addition, a USB serial port adapter board is required. The figure below shows one of the adapter boards.

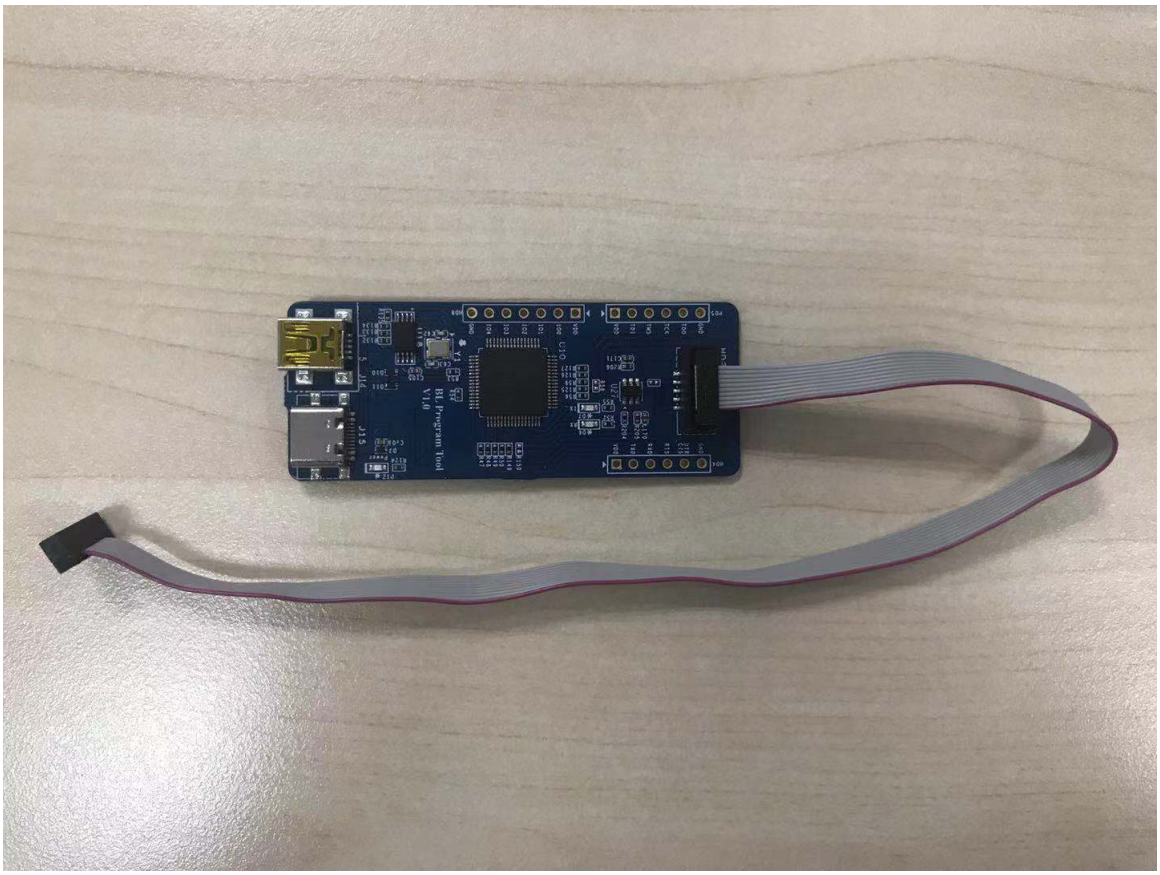

Fig. 4.2: Adapter board

The adapter board uses the USB interface for power supply. After connecting to the evaluation kit, it is used for UART download, UART print and online debug.

When the adapter board is connected to the PC, two serial ports will appear in the PC’s device manager. The two COM numbers are adjacent, and the smaller one is connected to the UART of the chip. If the serial port driver is not automatically installed, please go to https://www.ftdichip.com/Drivers/VCP.htm to download the driver.

After the adapter board is connected to the evaluation kit, run BLDevCube.exe, select BL702/704/706 in Chip Selection, then the following programming interface will be shown.

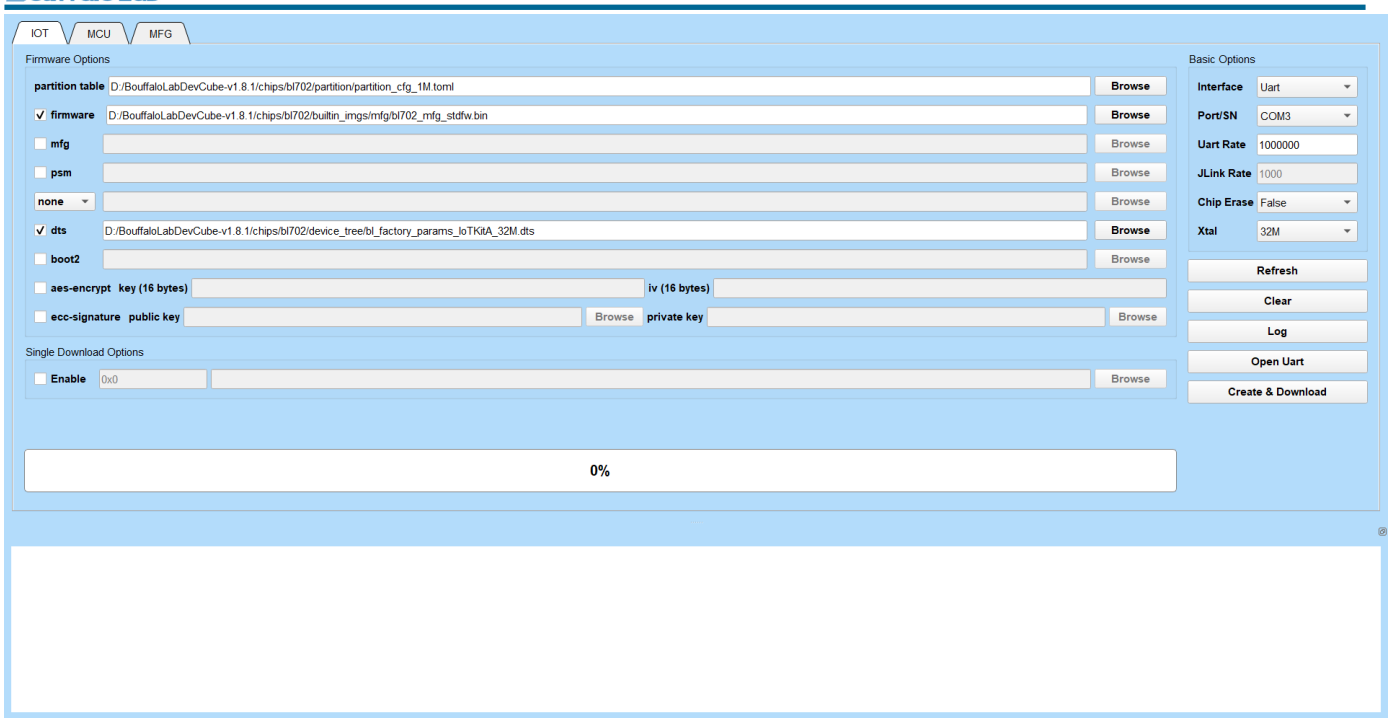

Fig. 4.3: Programming interface

In the communication interface settings on the right:

- Interface: Used to select the communication interface for programming, here select Uart for programming
- Port/SN: When selected Uart for programming, select the COM port that connects to the chip, and you can click the Refresh button to refresh the COM number
- Uart Rate: When selected Uart for programming, fill in the baud rate, 1000000 is recommended
- Chip Erase: Used to select whether full chip Flash will be erased before programming, default is False
- Xtal: Used to select the crystal type on the board, here select 32M

Use the default configuration for other items.

In the programming settings on the left, select:

- partition table: Use the partition table in the partition directory of the corresponding chip type in the programming tool directory. In this example bl702/partition/partition_cfg_1M.toml
- firmware: Select the corresponding Flash version firmware in the mfg folder under the builtin_imgs directory of the corresponding chip type in the programming tool directory. In this example bl702/builtin_imgs/mfg/bl702_mfg_-stdfw.bin
- dts: Use the device tree in the device_tree directory of the corresponding chip type in the programming tool directory. In this example bl702/device_tree/bl_factory_params_IoTKitA_32M.dts

According to the above configuration, after setting up Dev Cube, configure the chip to UART boot mode, and then

you can start programming.

The method to configure the chip to UART boot mode is as follows:

- Press and hold the Boot button on the module
- Press and release the RST button
- Release the Boot button

After completing the above boot settings, click the Create & Download button to complete the firmware programming. The indication of successful programming is as follows.

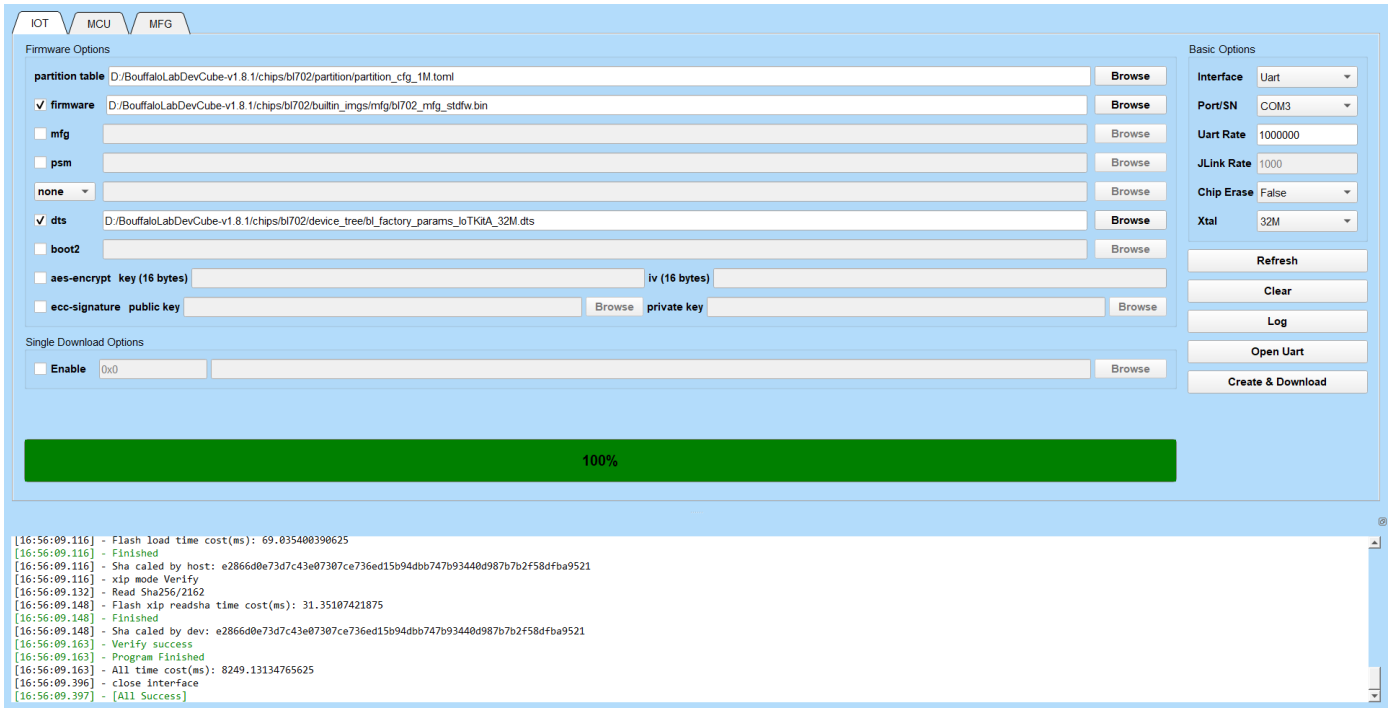

Fig. 4.4: Interface of successful programming

4.2 Program test firmware to BL702L/704L IOT module (with Flash)

The following takes BL702L IOT module evaluation kit as an example to introduce the programming process.

The BL702L IOT module evaluation kit BL702L_DVK is shown in the figure below.

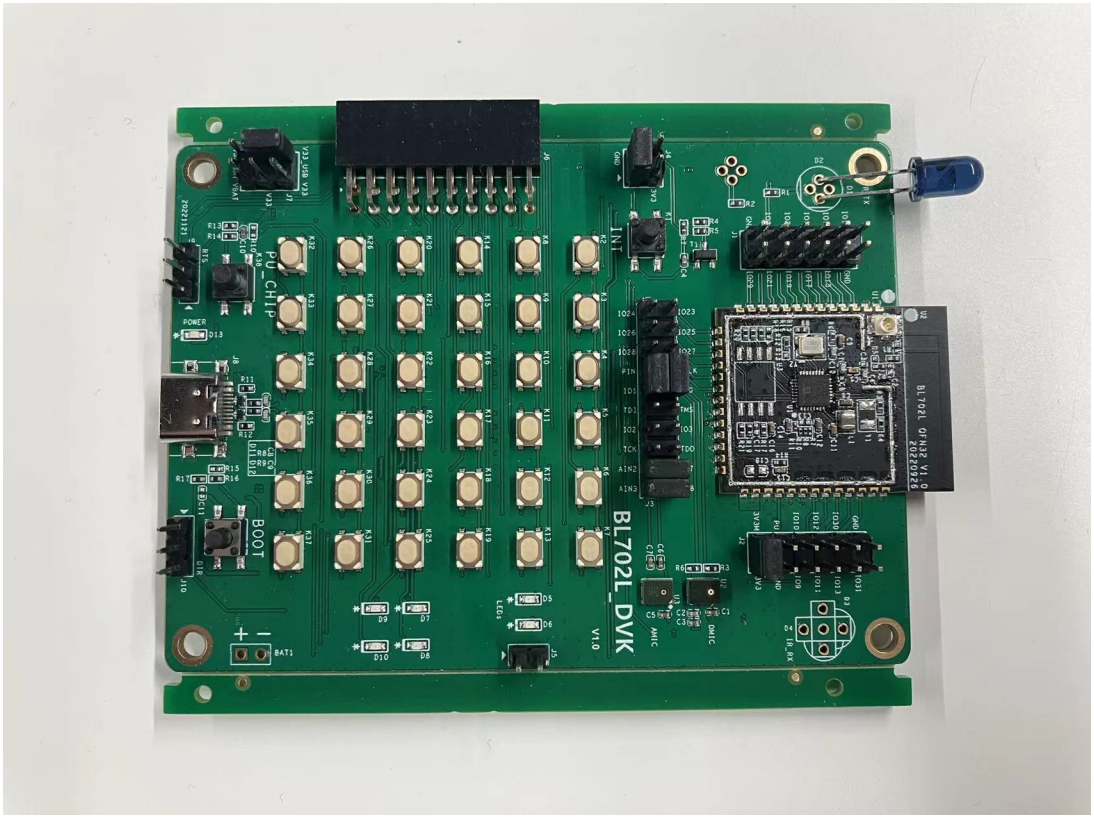

Fig. 4.5: IOT Module Evaluation Kit

The evaluation kit consists of an IOT module and a motherboard. The motherboard uses the USB interface for power supply.

When the evaluation kit is connected to the PC, one serial port and one CKLink-Lite will appear in the PC’s device manager, used for UART download, UART print and online debug.

Run BLDevCube.exe, select BL702L/704L in Chip Selection, then the following programming interface will be shown.

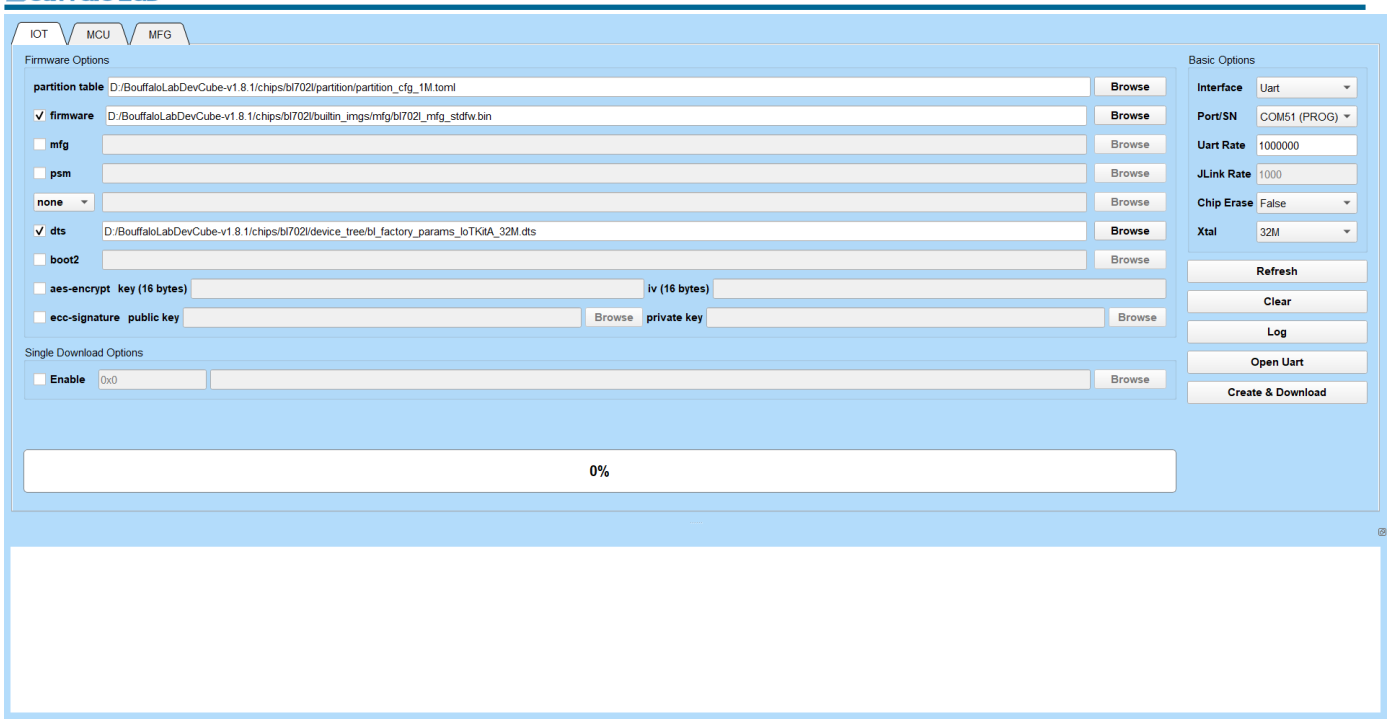

Fig. 4.6: Programming interface

In the communication interface settings on the right:

- Interface: Used to select the communication interface for programming, here select Uart for programming
- Port/SN: When selected Uart for programming, select the COM port that connects to the chip, and you can click the Refresh button to refresh the COM number
- Uart Rate: When selected Uart for programming, fill in the baud rate, 1000000 is recommended
- Chip Erase: Used to select whether full chip Flash will be erased before programming, default is False
- Xtal: Used to select the crystal type on the board, here select 32M

Use the default configuration for other items.

In the programming settings on the left, select:

- partition table: Use the partition table in the partition directory of the corresponding chip type in the programming tool directory. In this example bl702l/partition/partition_cfg_1M.toml
- firmware: Select the corresponding Flash version firmware in the mfg folder under the builtin_imgs directory of the corresponding chip type in the programming tool directory. In this example bl702l/builtin_imgs/mfg/bl702l_mfg_-stdfw.bin
- dts: Use the device tree in the device_tree directory of the corresponding chip type in the programming tool directory. In this example bl702l/device_tree/bl_factory_params_IoTKitA_32M.dts

According to the above configuration, after setting up Dev Cube, configure the chip to UART boot mode, and then

you can start programming.

The method to configure the chip to UART boot mode is as follows:

- Press and hold the BOOT button on the module
- Press and release the PU_CHIP button
- Release the BOOT button

After completing the above boot settings, click the Create & Download button to complete the firmware programming. The indication of successful programming is as follows.

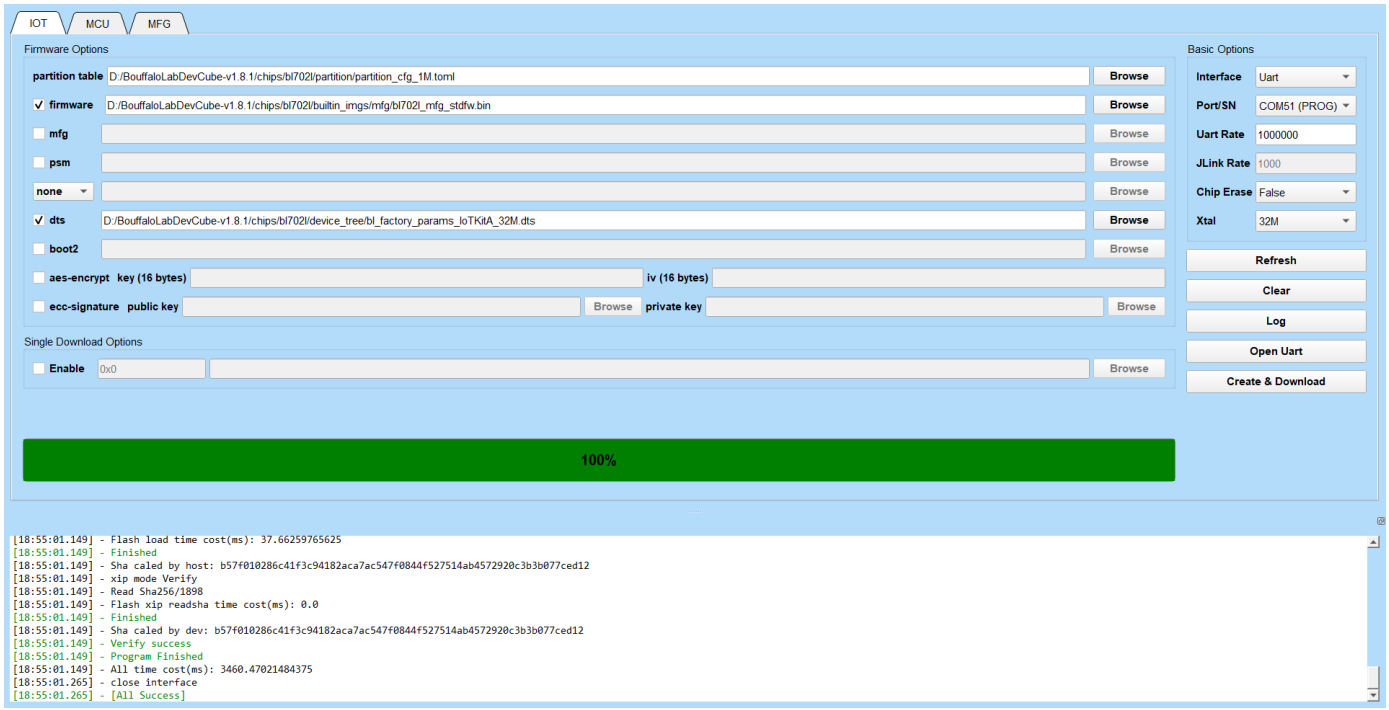

Fig. 4.7: Interface of successful programming

5 Run test firmware

For the IOT module (with Flash), after downloading the test firmware, reset the chip to run the RF test firmware.

On the BLDevCube.exe interface, switch to the RF MFG test interface through MFG Tab. Select the COM number used and click the Open Uart button to see the log of the test firmware. The example is as follows.

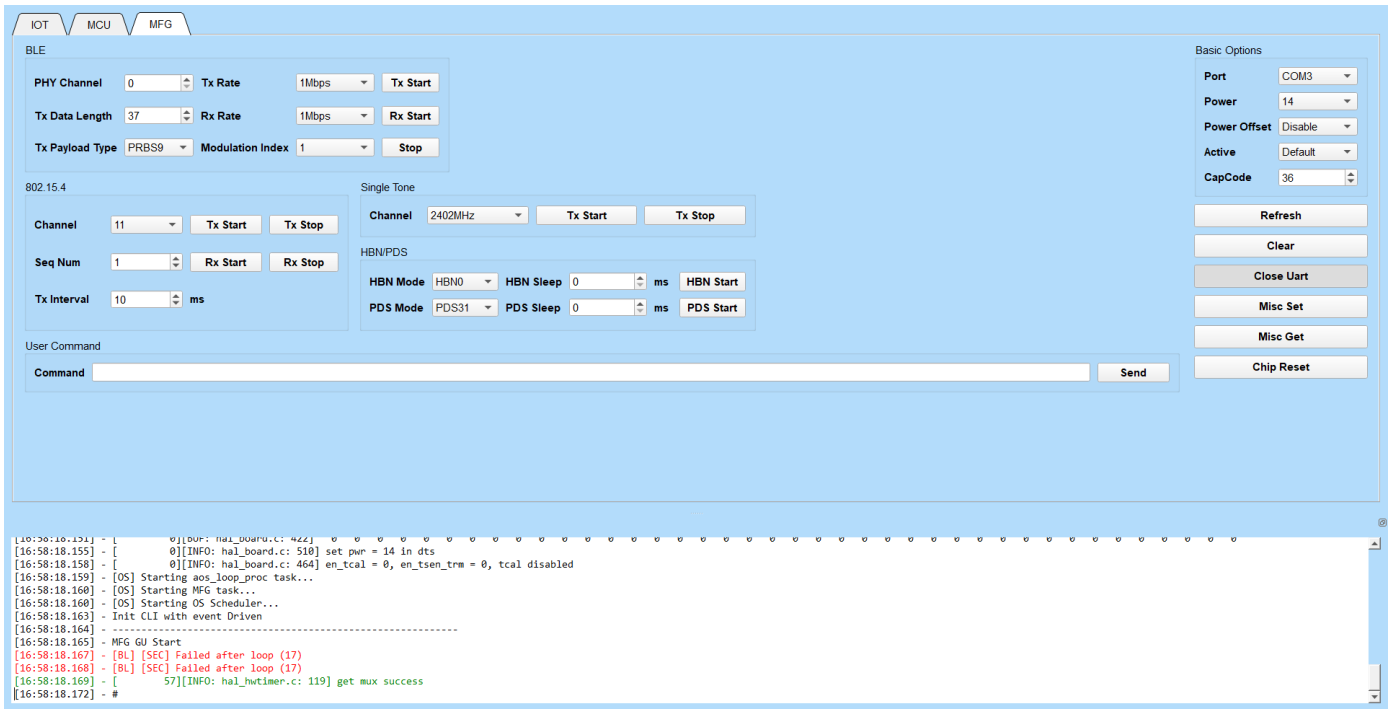

Fig. 5.1: Log of successful operation

The RF test tool running on PC communicates with the test firmware through UART, the baud rate used is 115200, the data bit is 8 bits, and there is no parity.

6 Basic configuration

Click the Misc Get button to get Power and CapCode parameters currently configured in the chip.

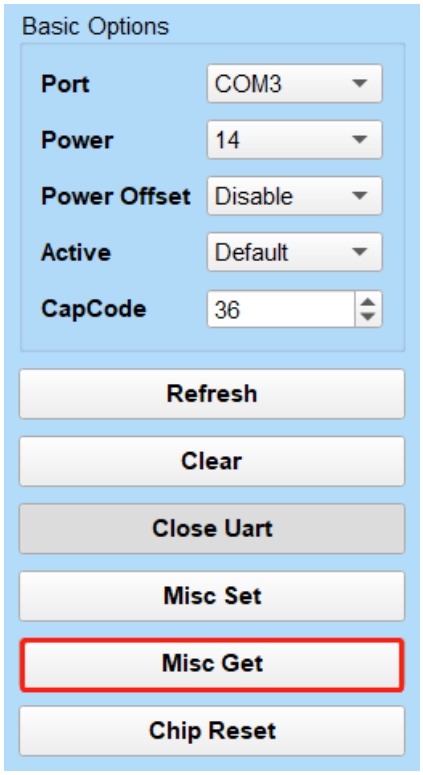

Fig. 6.1: Get parameters

Through the Power drop-down menu, you can set the tx power of the chip, and click the Misc Set button to take effect.

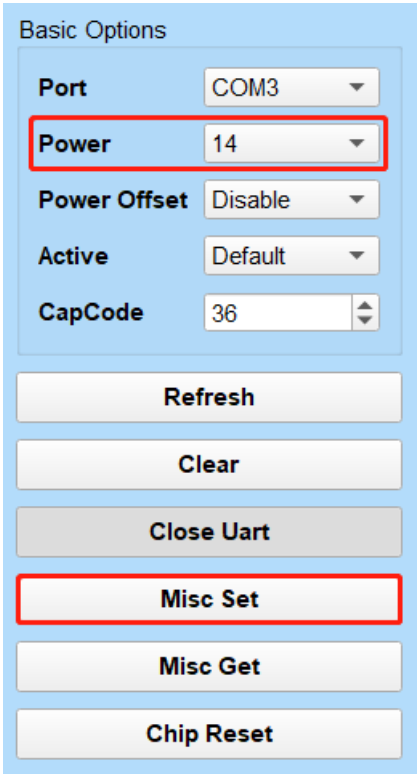

Fig. 6.2: Set tx power

Through the Power Offset drop-down menu, you can enable or disable power offset function, and click the Misc Set button to take effect. After chip reset, the power offset table will be initialized. Each time you enable power offset function, the power offset table will be updated.

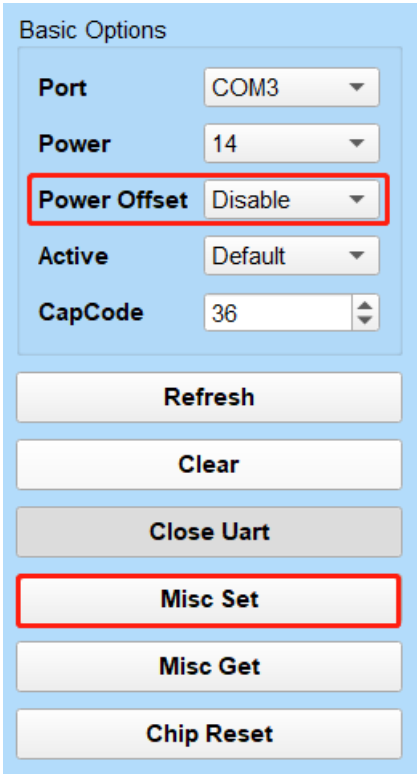

Fig. 6.3: Set power offset

For the load capacitance of the crystal, the chip has capacitance compensation inside. Different load capacitance requirements correspond to different capacitance compensation values. The following table provides reference values.

Note: Actual PCB traces also have certain parasitic capacitance, so the best compensation value is still subject to actual test results.

Table 6.1: Capacitance compensation value corresponding to BL702

XTAL Loading Capacity（pF）	Capacity Code
12	32~36

Table 6.2: Capacitance compensation value corresponding to BL702L

XTAL Loading Capacity（pF）	Capacity Code
12	113~123

The method of use is as follows:

1. Fill in the value to be compensated in the Cap Code.
2. Click the Misc Set button to update the compensation value.

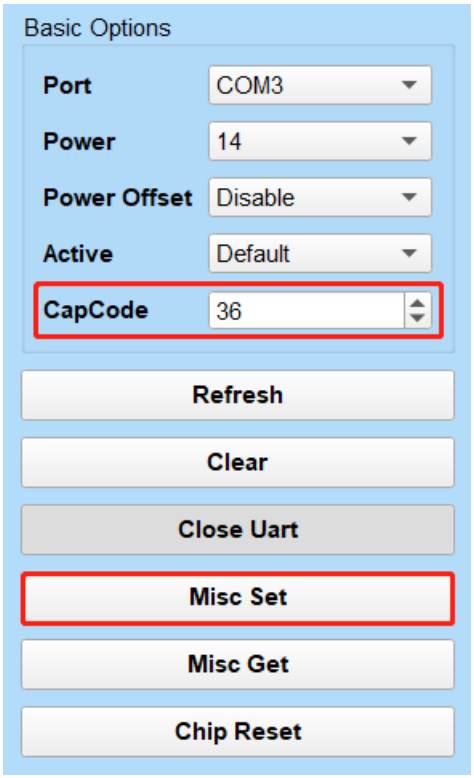

Fig. 6.4: Update capacitance compensation value

Through the Active drop-down menu, you can set the active mode of the chip, and click the Misc Set button to take effect. Among them, Idle mode will turn off more clocks and peripherals to achieve the purpose of reducing active power consumption.

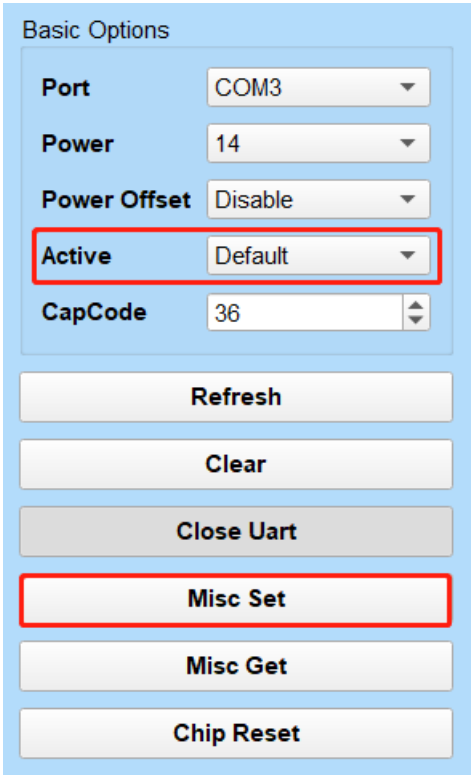

Fig. 6.5: Set active mode

7 Single Tone test

MFG supports Single Tone test mode. After setting the Channel, click Tx Start to start the test, and click Tx Stop to stop the test.

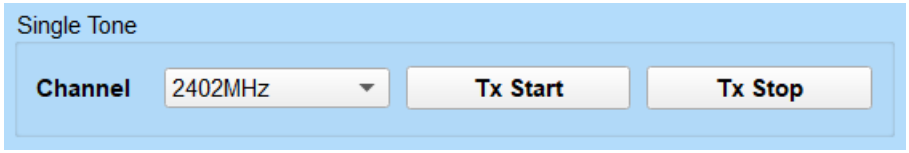

Fig. 7.1: Single Tone test

8 802.15.4 test

8.1 802.15.4 TX test

Set Channel for transmission channel, Seq Num for sequence number of the packet, and Tx Interval for packets tx interval.

Click Tx Start to start the test, and click Tx Stop to stop the test. After the test is over, the LOG area will show how many packets have been sent.

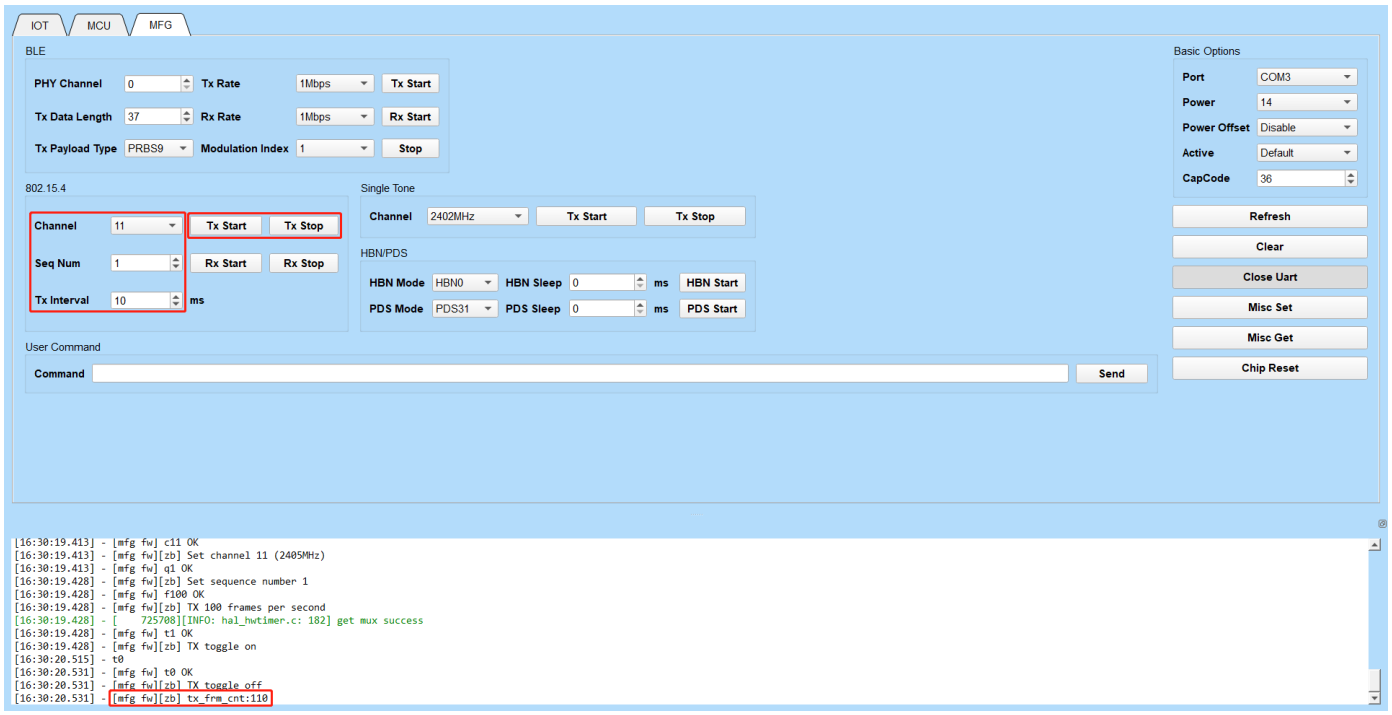

Fig. 8.1: 802.15.4 TX test

8.2 802.15.4 RX test

Set Channel for receiving channel, and Seq Num for sequence number of the packet.

Click Rx Start to start the test, and click Rx Stop to stop the test. After the test is over, the LOG area will show how many packets (both seq-num-matched packets and total packets) have been received, as well as the average rssi, average frequency offset, and average lqi.

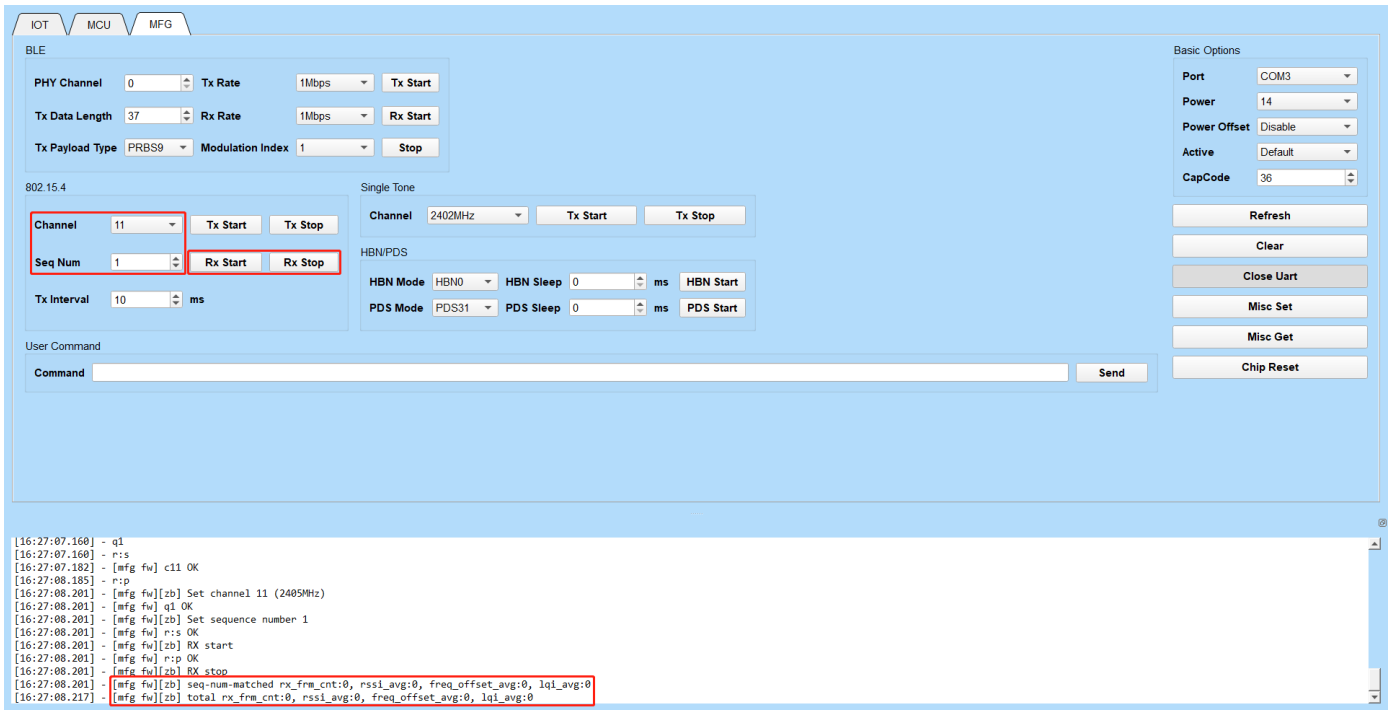

Fig. 8.2: 802.15.4 RX test

9 BLE test

RF MFG provides TX and RX testing of BLE.

For TX test, you can set the PHY Channel, Tx Data Length, Tx Payload Type and Tx Rate, and then click the Tx Start button to start the test.

For RX test, you can set the PHY Channel, Rx Rate and Modulation Index, and then click the Rx Start button to start the test.

The test can be stopped by the Stop button.

Note: The MFG firmware supports standard HCI command. Instruments that supports HCI command (e.g. R&S CMW500) can be connected for test.

The Tx Payload Type is shown in the figure below.

Value	Parameter Description
0x00	PRBS9 sequence '11111111100000111101…' (in transmission order) as described in [Vol 6] Part F, Section 4.1.5
0x01	Repeated '11110000' (in transmission order) sequence as described in [Vol 6] Part F, Section 4.1.5
0x02	Repeated '10101010' (in transmission order) sequence as described in [Vol 6] Part F, Section 4.1.5
0x03	PRBS15 sequence as described in [Vol 6] Part F, Section 4.1.5
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

Fig. 9.1: BLE Tx Payload Type

The Tx Rate is shown in the figure below.

PHY: *Size: 1 octet*

Value	Parameter Description
0x01	Transmitter set to use the LE 1M PHY
0x02	Transmitter set to use the LE 2M PHY
0x03	Transmitter set to use the LE Coded PHY with S=8 data coding
0x04	Transmitter set to use the LE Coded PHY with S=2 data coding
All other values	Reserved for future use

Fig. 9.2: BLE Tx Rate

The Rx Rate is shown in the figure below.

PHY: *Size: 1 octet*

Value	Parameter Description
0x01	Receiver set to use the LE 1M PHY
0x02	Receiver set to use the LE 2M PHY
0x03	Receiver set to use the LE Coded PHY
All other values	Reserved for future use

Fig. 9.3: BLE Rx Rate

The Modulation Index is shown in the figure below.

Modulation_Index: *Size: 1 octet*

Value	Parameter Description
0x00	Assume transmitter will have a standard modulation index
0x01	Assume transmitter will have a stable modulation index
All other values	Reserved for future use

Fig. 9.4: BLE Modulation Index

10 HBN/PDS test

The HBN/PDS test allows the chip to enter a low power mode. In HBN mode, only a small part of the circuit is kept in working state, while the power supply of other circuits is turned off, thus the power consumption reaches the lowest level.

The chip can be waken up from HBN/PDS mode, and the chip will restart after waken up. The current test tool only supports RTC wakeup.

After setting the HBN/PDS sleep time, click the corresponding Start button to let the chip enter the HBN/PDS mode. After the sleep time expires, the chip will restart.

If the sleep time is set to 0, it means permanent sleep.

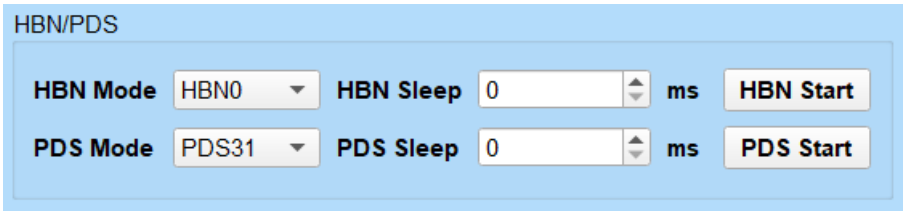

Fig. 10.1: HBN/PDS parameter setting

11

Programming guide for mass production test

For mass production test, both RF test firmware and user firmware may need programming. After programming, the RF test firmware will run after chip reset, and instruments (e.g. iTest) can be connected for mass production test. After mass production test completes, the user firmware will run after chip reset. If mass production fails, the RF test firmware will still run after chip reset.

The programming procedure is as follow:

1. Select a partition table file which is used for IOT downloading interface, e.g. partition_cfg_1M.toml, then modify the “activeindex” of FW entry to 1, which means booting from FW1, namely booting from mfg, because FW1 and mfg are of the same entry.

```
[pt_table]
#partition table is 4K in size
address0 = 0x1000
address1 = 0x2000

[[pt_entry]]
type = 0
name = "FW"
device = 0
address0 = 0x3000
size0 = 0x90000
address1 = 0x93000
size1 = 0x66000
# compressed image must set len,normal image can left it to 0
len = 0
activeindex = 0
age = 0

[[pt_entry]]
type = 1
name = "mfg"
device = 0
address0 = 0x93000
size0 = 0x66000
address1 = 0
size1 = 0
# compressed image must set len,normal image can left it to 0
len = 0
activeindex = 0
age = 0
```

0 -> 1

Fig. 11.1: Modify active entry

2. In IOT downloading interface, select the partition table file modified in the previous step as Partition Table, select a user firmware as Firmware Bin, select the RF test firmware as MFG Bin, and keep others unchanged.

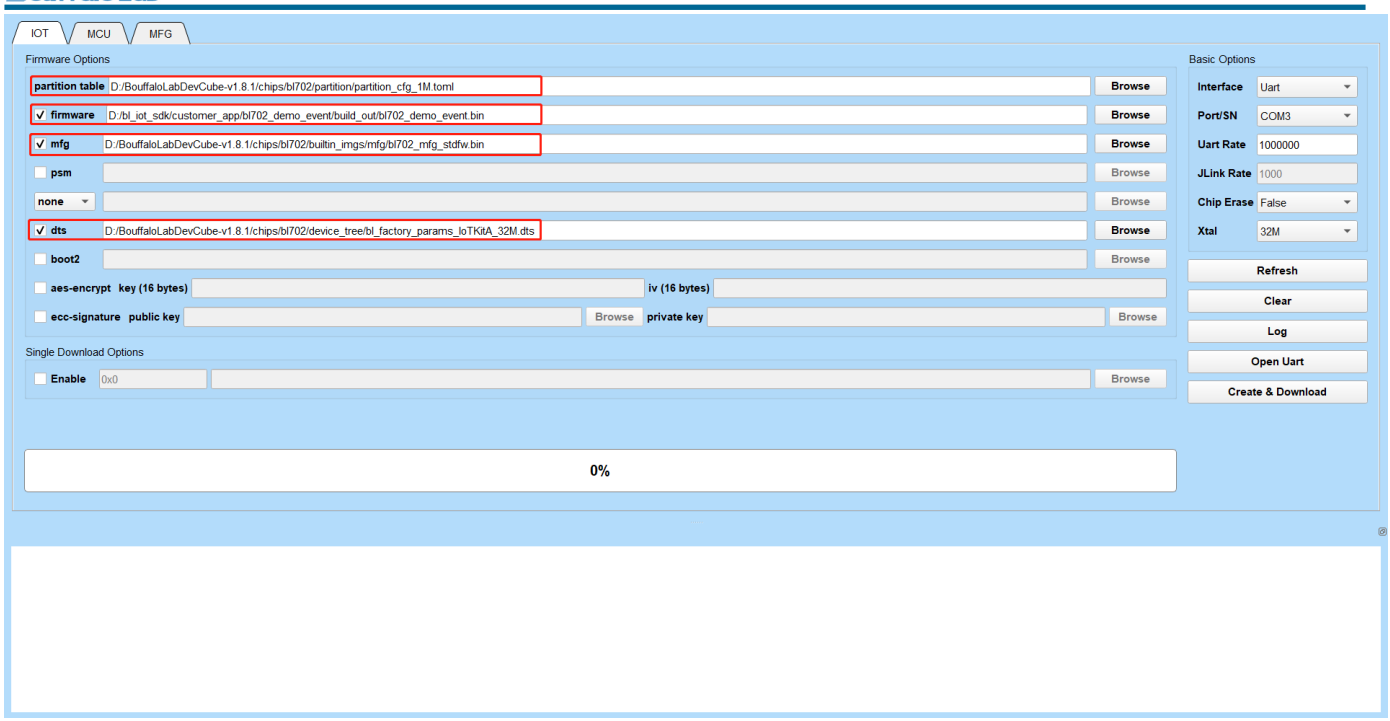

Fig. 11.2: Programming interface for mass production test

To exit mass production test：

1. The host send the ATSC command to set user firmware entry as active entry. If the user firmware is not programmed, the ATSC command will return an error.
2. The host send the Reset command to reset the chip. After chip reset, the user firmware will run.

12 Serial communication commands

All commands are of string type.

12.1 Shakehand

- Command: `H`
- Return: `mfg`

12.2 Get cap code

- Command: `x`
- Return: `Get cap code [cap code]`

12.3 Set cap code

- Command: `x[cap code]`

cap code (BL702): [0, 63]; default: 36

cap code (BL702L): [0, 255]; default: 118

12.4 Get power

- Command: `p`
- Return: `Get tx power [power dbm]dBm`

12.5 Set power

- Command: `p[power dbm]`

power dbm (BL702): [0, 14]; default: 14

power dbm (BL702L): [0, 10]; default: 10

12.6 Set channel

- Command: `c[channel]`

For single tone test, channel is the channel frequency - channel: [2402, 2480]; default: 2402

For 802.15.4 test, channel is the channel index number - channel: [11, 26]; default: 11

12.7 Set TX frequency

- Command: `f[freq]`

freq is the number of packets sent per second

freq: [1, 1000]; default: 100

12.8 Set sequence number

- Command: `q[seq num]`

seq num: [0, 255]; default: 1

12.9 TX

1. Tx Start: `t1`
2. Tx Stop: `t0`

- Return: `tx_frm_cnt:[tx_frm_cnt]`

12.10 RX

1. Rx Start: `r:s`
2. Rx Stop: `r:p`

- Return: `rx_frm_cnt:[rx_frm_cnt], rssi_avg:[rssi_avg], freq_offset_avg:[freq_offset_avg], lqi_avg:[lqi_avg]`

12.11 HBN

1. set hbn level: `hl[hbn level]`
2. enter hbn mode: `ht[sleep time ms]`

hbn level: [0, 2]; default: 0

sleep time ms: [0, 131071999]; default: 0(sleep forever)

12.12 PDS

1. set pds level: `sl[pds level]`
2. enter pds mode: `st[sleep time ms]`

pds level: 31; default: 31

sleep time ms: [0, 131071999]; default: 0(sleep forever)

12.13 Power Offset

1. Enable: `V1`
2. Disable: `V0`

12.14 Single Tone

1. Tx Start: `m1`
2. Tx Stop: `m0`

12.15 CCA

1. set ed threshold: `C:T[ed threshold]`
2. run cca: `C:M[cca mode]`

- Return: `channel_busy:[channel_busy], rssi:[rssi], ed:[ed]`

ed threshold: [-122, 5]; default: -71

cca mode: [0, 3]; default: 2

12.16 Active

1. Default: `i0`
2. Idle: `i1`

12.17 Read memory

- Command: `RM0x[addr]`
- Return: `Read memory: 0x[addr] = 0x[val]`

12.18 Write memory

- Command: `SM0x[addr]=0x[val]`
- Return: `Write memory: 0x[addr] = 0x[val]`

12.19 Write cap code to efuse register

- Command: `ewx[cap code]`
- Return: `Write cap code [cap code] to efuse register`

cap code (BL702): [0, 63]

cap code (BL702L): [0, 255]

Note: The ewx command will first reload all data from efuse to corresponding efuse registers, then write cap code to corresponding efuse register, without limit of write times.

12.20 Write power offset to efuse register (BL702)

- Command: `ewp[power offset low],[power offset high]`
- Return: `Write power offset [power offset low],[power offset high] to efuse register`

power offset low: [-8, 7]

power offset high: [-8, 7]

12.21 Write power offset to efuse register (BL702L)

- Command: `ewp[power offset 1],[power offset 2],[power offset 3],[power offset 4]`
- Return: `Write power offset [power offset 1],[power offset 2],[power offset 3],[power offset 4] to efuse register`

power offset 1: [-8, 7]

power offset 2: [-8, 7]

power offset 3: [-8, 7]

power offset 4: [-8, 7]

Note: The ewp command will first reload all data from efuse to corresponding efuse registers, then write power offset to corresponding efuse register, without limit of write times.

12.22 Write mac address to efuse register

- Command: `ewm[mac0]:[mac1]:[mac2]:[mac3]:[mac4]:[mac5]:[mac6]:[mac7]`
- Return: `Write mac [mac0]:[mac1]:[mac2]:[mac3]:[mac4]:[mac5]:[mac6]:[mac7] to efuse register`

Example: ewm00:00:75:09:00:42:E8:B4

Note: The ewm command will first reload all data from efuse to corresponding efuse registers, then write mac address to corresponding efuse register, without limit of write times.

12.23 Read cap code from efuse register

- Command: `erx`
- Return: `Read cap code [cap code] from efuse register`

12.24 Read power offset from efuse register

- Command: `erp`
- Return: `Read power offset [power offset low],[power offset high] from efuse register`

12.25 Read mac address from efuse register

- Command: `erm`
- Return: `Read mac [mac0]:[mac1]:[mac2]:[mac3]:[mac4]:[mac5]:[mac6]:[mac7] from efuse register`

12.26 Program efuse

- Command: `ep`

Note: The ep command will write all data in efuse registers to efuse permanently, i.e. ewx+ep will write cap code to efuse, ewp+ep will write power offset to efuse, and ewm+ep will write mac address to efuse. The programming times of cap code, power offset and mac address for each chip is shown below:

BL702: 3 times for cap code, 3 times for power offset, 3 times for mac address

BL702L: 3 times for cap code, 3 times for power offset, 2 times for mac address

---

12.27 Cap code temperature calibration (BL702L)

1. Enable: `X1`
2. Disable: `X0`

Note: When enabled, current cap code register value will be stored, and cap code in efuse will be loaded into cap code register. When disabled, previous stored cap code register value will be restored.

12.28 Get MFG FW version

- Command: `y:v`
- Return: `MFG Version: [version]`

12.29 Get MFG FW build infomation

- Command: `y:d`
- Return: `Build Data: [build date] Build Time: [build time]`

12.30 Get temperature

- Command: `y:T`
- Return: `Temperature: [temperature]`

12.31 Get MAC address

- Command: `y:m`
- Return: `MAC Address: [mac address]`

12.32 Exit MFG FW

- 命令：`ATSC`

Note: The ATSC command will set user firmware entry as active entry. After next chip reset, the user firmware will run. If the user firmware is not programmed, the ATSC command will return an error.

12.33 Reset chip

- Command: `Reset`

12.34 BLE Test

12.34.1 BLE TX

- Command: `ETE[channel][tx data length][tx payload type][tx rate]`

All parameters are of hex string type.

Example: ETE00250001

channel: 0x00

tx data length: 0x25

tx payload type: 0x00(PRBS9)

tx rate: 0x01(1Mbps)

12.34.2 BLE RX

- Command: `ERE[channel][rx rate][modulation index]`

All parameters are of hex string type.

Example: ERE000100

channel: 0x00

rx rate: 0x01(1Mbps)

modulation index: 0x00

12.34.3 BLE test stop

- Command: `EE`