EC2x&EG25-G QuecOpen NF3303 BT Application Note

LTE Standard Module Series

Version: 1.0.0

Date: 2020-12-28

Status: Preliminary

Our aim is to provide customers with timely and comprehensive service. For any assistance, please contact our company headquarters:

Quectel Wireless Solutions Co., Ltd.
Building 5, Shanghai Business Park Phase III (Area B), No.1016 Tianlin Road, Minhang District, Shanghai 200233, China
Tel: +86 21 5108 6236
Email: info@quectel.com

Or our local office. For more information, please visit:
http://www.quectel.com/support/sales.htm.

For technical support, or to report documentation errors, please visit:
http://www.quectel.com/support/technical.htm
Or email to support@quectel.com.

General Notes

Quectel offers the information as a service to its customers. The information provided is based upon customers' requirements. Quectel makes every effort to ensure the quality of the information it makes available. Quectel does not make any warranty as to the information contained herein, and does not accept any liability for any injury, loss or damage of any kind incurred by use of or reliance upon the information. All information supplied herein is subject to change without prior notice.

Disclaimer

While Quectel has made efforts to ensure that the functions and features under development are free from errors, it is possible that these functions and features could contain errors, inaccuracies and omissions. Unless otherwise provided by valid agreement, Quectel makes no warranties of any kind, implied or express, with respect to the use of features and functions under development. To the maximum extent permitted by law, Quectel excludes all liability for any loss or damage suffered in connection with the use of the functions and features under development, regardless of whether such loss or damage may have been foreseeable.

Duty of Confidentiality

The Receiving Party shall keep confidential all documentation and information provided by Quectel, except when the specific permission has been granted by Quectel. The Receiving Party shall not access or use Quectel's documentation and information for any purpose except as expressly provided herein. Furthermore, the Receiving Party shall not disclose any of the Quectel's documentation and information to any third party without the prior written consent by Quectel. For any noncompliance to the above requirements, unauthorized use, or other illegal or malicious use of the documentation and information, Quectel will reserve the right to take legal action.

Copyright

About the Document

Revision History

Version	Date	Author	Description
-	2020-08-17	Arthur CHEN	Creation of the document
1.0.0	2020-12-28	Arthur CHEN	Preliminary

Contents

Table Index

Figure Index

1 Introduction

Quectel LTE Standard modules support QuecOpen® solution. QuecOpen is an open-source embedded development platform based on Linux system. It is intended to simplify the design and development of IoT applications.

The document is designed to quickly use the Bluetooth of NF3303 module (Hereinafter referred to as “NF3303”) on Quectel QuecOpen® modules (Hereinafter referred to as “the module”) to achieve Bluetooth related services.

The applicable QuecOpen modules include:

- EC20 R2.1 QuecOpen
- EC21 Series QuecOpen
- EC25 Series QuecOpen
- EG25-G QuecOpen

2 Hardware Information

NF3303 and the module communicate through the four-wire serial port which supports hardware flow control. Besides, the module should reserve another GPIO pin as the power control pin of NF3303. However, to enable the sleep and wake-up function, the module needs to reserve two additional pins: one pin serves as an input interrupt; the other one as a common GPIO output pin.

The correspondence between the pins of the module and NF3303 is shown in the following table:

Table 1: Correspondence between Pins of the Module and NF3303

Quectel Module		NF3303 Module	
Pin No.	**Definition**	**Pin No.**	**Definition**
2	GPIO_10(Output)	40	BT_DEVWAKE(Input)
3	GPIO_42(Input)	39	BT_HOST_WAKE_UP(Output)
139	PMU_GPIO1019(Output)	24	BT_EN(Input)
37	UART_RTS_BLSP6/GPIO_22(Input)	10	BT_UART_RTS_N(Output)
38	UART_RXD_BLSP6/GPIO_21(Input)	7	BT_UART_TXD(Output)
39	UART_TXD_BLSP6/GPIO_20(Output)	6	BT_UART_RXD(Input)
40	UART_CTS_BLSP6/GPIO_23(Output)	11	BT_UART_CTS_N(Input)
26	PCM_SYNC/GPIO_79(Output)	34	AUDIO_FSYNC_BT(Input)
27	PCM_CLK/GPIO_78(Output)	33	AUD_CLK_BT(Input)
25	PCM_OUT/GPIO_77(Output)	32	AUD_IN_BT(Input)
24	PCM_IN/GPIO_76(Input)	35	AUD_OUT_BT(Output)

3 Bluetooth Function Configuration

3.1. Match the Pin

According to ***Table 1***, BLSP needs to be multiplexed as UART, and that BT_EN pin should be set as GPIO output, BT_HOST_WAKE_UP pin as interrupt input, and BT_DEVWAKE as GPIO output.

3.1.1. Configure BLSP to uart6

The module has 6 BLSP controllers, all of which can be multiplexed as UART/I2C/SPI peripheral interfaces. It should be noted that UART/I2C/SPI peripheral device interfaces cannot be used simultaneously. These interfaces support RTS/CTS. If BLSP is multiplexed as uart6 for Bluetooth communication, please disable spi_6 and enable blsp1_uart6 in the *mdm9607-mtp.dtsi* file.

```
&spi_6 {
-     status = "ok";
+     status = "disabled";
};
&blsp1_uart6 {
-     status = "disabled";
+     status = "ok";
};
```

3.1.2. Set BT_EN Pin

QuecOpen SDK supports configuring the module pins as common GPIOs through the application layer for data output. Please configure the GPIOs according to the module's hardware design.

The module needs to pull BT_EN to up/down according to the power-on timing requirements of NF3303. The specific process is: pull down BT_EN → hold for 200 seconds → pull up BT_EN. Please modify the functions of *ql_bt_en_pin_init* and *ql_bt_module_enable* in the file *ql_nf3303_ble_common.c* in the path of *ql-ol-sdk/ql-ol-extsdk/example/bt/nf3303/source*. See the example as below:

```
/*Operate BT_EN to enable BT Module*/
void ql_bt_module_enable()
{
    /*Pull Up PM_ENABLE to Power the Module*/
    system("echo 1 > /sys/kernel/debug/regulator/rome_vreg/enable");
    /*Reset BT Module*/
    ioctl(bt_power_fd, 0);    //pull Down BT_EN
    usleep(200000);
    ioctl(bt_power_fd, 1);    //pull Up BT_EN
}
```

3.1.3. Set BT_HOST_WAKE_UP Pin

QuecOpen SDK supports configuring the module pins as interrupts through the application layer. Please configure the GPIOs according to the module's hardware design. Please modify the macro definition BT_HostWakePin of file *ql_NF3303_ble_sleep.c* in *ql-ol-sdk/ql-ol-extsdk/example/bt/nf3303/source* path to the pins used by the module's hardware. Please see the example as follows:

```
#if defined(__QUECTEL_PROJECT_AG35C__) || defined(__QUECTEL_PROJECT_AG35CE__) \
    || defined(__QUECTEL_PROJECT_AG35CEN__)|| defined(__QUECTEL_PROJECT_AG35CEVBM__)
static Enum_PinName BT_HostWakePin = PINNAME_GPIO3;
static Enum_PinName BT_DevWakePin = PINNAME_GPIO6;
#else
static Enum_PinName BT_HostWakePin = PINNAME_GPIO3;
static Enum_PinName BT_DevWakePin = PINNAME_GPIO2;
#endif
```

3.1.4. Set BT_DEVWAKE Pin

QuecOpen SDK supports configuring the module pins as common GPIOs through the application layer for data output. Please configure the GPIOs according to the module's hardware design. Please modify the macro definition *BT_DevWakePin* of file *ql_NF3303_ble_sleep.c* in the path of *ql-ol-sdk/ql-ol-extsdk/example/bt/nf3303/source* to the pins used by the module's hardware. Please see the example as follows:

```
#if defined(__QUECTEL_PROJECT_AG35C__) || defined(__QUECTEL_PROJECT_AG35CE__) \
    || defined(__QUECTEL_PROJECT_AG35CEN__)|| defined(__QUECTEL_PROJECT_AG35CEVBM__)
static Enum_PinName BT_HostWakePin = PINNAME_GPIO3;
static Enum_PinName BT_DevWakePin = PINNAME_GPIO6;
#else
static Enum_PinName BT_HostWakePin = PINNAME_GPIO3;
static Enum_PinName BT_DevWakePin = PINNAME_GPIO2;
#endif
```

3.2. Function of Waking up from Sleep

3.2.1. Pins of Waking up from Sleep

NF3303 uses two pins to realize the function of waking up from sleep, respectively:

- BT_HOST_WAKE_UP: This pin is an interrupt input pin for the module; it is an output pin for NF3303.
- BT_DEVWAKE: This pin is a common output pin for the module; it is an input pin for NF3303.

3.2.1.1. BT_HOST_WAKE_UP Function

When there is no data interaction, BT_HOST_WAKE is in low level, allowing the module to sleep. When the mobile phone sends a request of reading and writing data, BT_HOST_WAKE_UP is pulled high by NF3303, that is, a rising edge interrupt occurs to the module to wake up the module. After the data is sent, BT_HOST_WAKE_UP is pulled low (a falling edge) by NF3303 and the module goes to sleep again.

3.2.1.2. BT_DEVWAKE Function

When the module sends data to NF3303, the Bluetooth protocol stack generates a callback function, and the module pulls the BT_DEV_WAKE pin high according to the callback function, allowing NF3303 to sleep. After the data is sent, the Bluetooth protocol stack also generates a callback function, and the module pulls the BT_DEV_WAKE pin low according to the callback function and NF3303 goes to sleep again.

NOTE

If NF3303 needs to transfer data to the module, the basis to determine whether it can send data to the module is the CTS level, that is, if RTS level of the module is low, NF3303 sends data to the module. Please ensure that the module's RTS output is high in the sleep state to avoid data loss.

3.2.2. Coding Logic of Waking up the Module from Sleep

3.2.2.1. Initial Phase of Waking up from Sleep

After waking up the module by pulling BT_HOST_WAKE_UP pin, the wakelock needs to be enabled to prevent the module from sleeping again. The module sleeps again after releasing the wakelock.

3.2.2.2. Receive BT_HOST_WAKE_UP Rising Edge Interrupt Code

Check whether the wakelock is locked, if it is locked, there is no need to lock it. If it is not locked, it needs to be locked.

3.2.2.3. Receive BT_HOST_WAKE_UP Falling edge Interrupt Code

When the module system is started, a periodic timer is started with an interval of 5 seconds. When the timer times out, the system checks whether it receives the BT_HOST_WAKE_UP falling edge interrupt, and then check whether the wakelock has been released. If it has been released, there is no need to release it; if it has not been unreleased, it is released to allow the module to sleep.

3.2.2.4. Code of BT_DEV_WAKE Pin

If the module receives callback function *NFBT_WARNING_BT_ALLOW_SLEEP* from the Bluetooth protocol stack, BT_DEVWAKE is pulled low. If it receives stack callback function *NFBT_WARNING_BT_WAKE* from the Bluetooth protocol, BT_DEVWAKE is pulled high.

3.3. Bluegate Protocol Stack

The Bluegate protocol stack provided by NF3303 manufacturer simplifies the Bluetooth function and can implement most Bluetooth functions. As a middleware, Bluegate implements a variety of commonly used protocol stacks and profiles, and provides a set of application programming interfaces (APIs). Please refer to ***document [1]*** to develop applications according to actual application requirements.

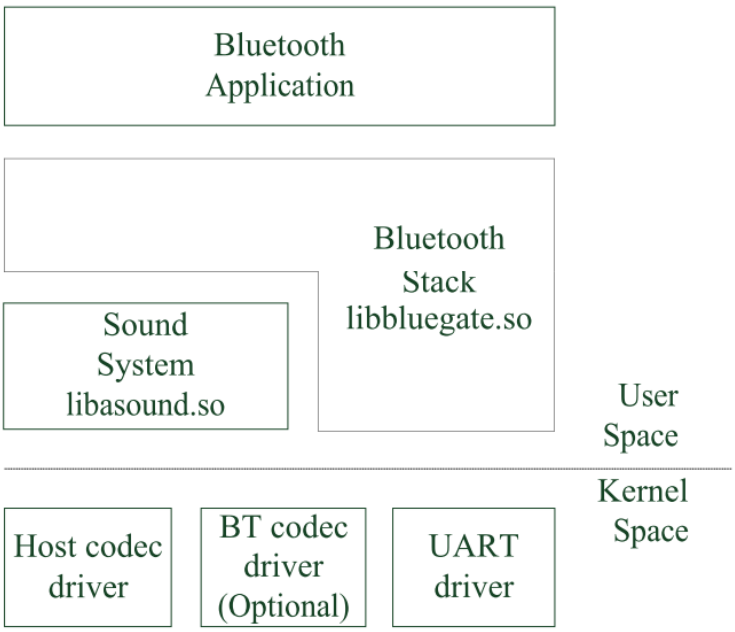


Figure 1: Bluegate Framework

3.3.1. Configure Communication Serial Port

After confirming that the serial driver has been configured correctly, import the file *bluegate_hw.conf* in the path *ql-ol-sdk/ql-ol-extsdk/example/bt/nf3303/lib/firmware* into the */etc/bluetooth/* directory. The *bluegate_hw.conf* file is shown below:

```
# UART device port where Bluetooth controller is attached
UartPort = /dev/ttyHSL1

# Firmware patch file location
FwPatchFilePath = /etc/bluetooth
FwPatchFileName = BCM4339_003.001.009.0119.0000.hcd
```

The text information of the modified configuration file is as follows:

```
# UART device port where Bluetooth controller is attached
UartPort = /dev/ttyHSL1

# Firmware patch file location
FwPatchFilePath = /etc/bluetooth/
FwPatchFileName = BCM4339_003.001.009.0119.0000.hcd
```

In it, *UartPort* must be consistent with the actual device information of the serial port in use. *FwPatchFilePath* must be consistent with the path where the actual file *BCM4339_003.001.009.0119.0000.hcd* is located.

3.3.2. Import NF3303 Firmware and Bluetooth Protocol Stack

3.3.2.1. Import NF3303 Firmware

After the serial communication with NF3303 is established successfully, the module transfers the firmware to NF3303 based on the **HCI** command, so NF3303 firmware needs to be placed in the file system in advance. ADB tool can be used to push the file *BCM4339_003.001.009.0119.0000.hcd* in the path of *ql-ol-sdk/ql-ol-extsdk/example/bt/nf3303/lib/firmware* to the */etc/bluetooth/* directory of the module, the reference command is as follows:

```
$ adb push BCM4339_003.001.009.0119.0000.hcd /etc/bluetooth/
```

3.3.2.2. Import Bluetooth Protocol Stack

The Bluetooth protocol stack (*libbluegate.so*) is provided in the form of a dynamic library, on which the application depends when it is running. Therefore, the Bluetooth protocol stack needs to be placed in the file system in advance. ADB tool can be used to push the file *libbluegate.so* in the path of *ql-ol-sdk/ql-ol-extsdk/example/bt/nf3303/lib/* to the */lib/* directory of the module, the reference command is as follows:

```
$ adb push libbluegate.so /lib/
```

4 Functional Introduction

The mobile phone needs to download dedicated software for functional verification. If it is an Android system, please download BLEDeng in the application store or search engine. If it is an iOS system, please download LightBlue in the application store. This chapter uses the BLEDeng under the Android system as an example for detailed introduction.

4.1. Preparation

4.1.1. Complile demo_UI

According to the SDK manual of the module, please obtain the latest SDK and complete the construction of the cross-compilation environment. Then enter the *ql-ol-sdk/ql-ol-extsdk/example/bt/nf3303* directory and execute the following command to generate the executable program demo_UI. After successful compilation, please copy to the rootfs file system.

```
# make clean
# make
```

```
edison@edison-VirtualBox:~/9X28/ql-ol-sdk_/ql-ol-extsdk/example/bt/nf3303$ ls
demo_UI  include  lib  Makefile  README  source
```

4.1.2. Run demo_UI

Enter the command **amix 'SEC_AUX_PCM_RX Audio Mixer MultiMedia1' 1** in the command line to run demo_UI, enter **help** to view the operation commands.

```
/data # ./demo_UI
Register CBK success


************************************************************
****** nFore BlueGate Bluetooth Middleware (v0.3.2.75) *****
************************************************************

Enable BT stack success

Enter the command to execute:
BT Adapter must reset hardware.

Get BT Userial fd: 16

Local device infos:
    Name:   [Bluegate]
    Addr:   [A4:04:50:30:26:31]
    Device Sleep


Wake BT controller right now!

bt state: enable

Local device infos:
    Name:   [Bluegate]
    Addr:   [A4:04:50:30:26:31]
    Device Connectable


Client Register status=0
Service Register status=0
Allow BT controller into sleep

help

****************************************** Help ******************************************
bt_enable(c1)           : Enable BT stack.
bt_disable(c2)          : Disable BT stack.
localinfos(c3)          : Get local device infos.
setname(c4)             : Set local device name.
setscan(c5)             : Set scan mode.
scan(c6)                : Scan nearby bluetooth devices.
stop(c7)                : Stop scan nearby bluetooth devices.
pairlist(c8)            : List paired remote device.
pair(c9)                : Pair with remote device.
delete(c10)             : Delete paired remote device.
connect(c13)            : Connect to paired remote device.
```

NOTE

Please manually input the command **amix 'SEC_AUX_PCM_RX Audio Mixer MultiMedia1' 1** when it is firstly executed.

4.2. Enable BLE Broadcast

Enter the commands **c345** -> **0** -> **c328** -> **c311** -> **1** -> **1** in sequence to enable the BLE function, as shown in the following figure:

```
getring(c215)                   : Get ringtone file path.
setringpth(c216)                : Get ringtone file path.
****************************************** Help End ******************************************

Enter the command to execute:

c345

Set GATT role to Peripheral(0), Central(1), or -1 to cancel.

0
Gatt set role : status=0, role=0

Enter the command to execute:
c328

Enter the command to execute:
Gatt service added : status=0 srvc_handle=40
                   : service UUID=000018FF00001000800000805F9B34FB
Gatt characteristic added status : status=0, srvc_handle=40, char. handle=42
                   : characteristic UUID=00002AFF00001000800000805F9B34FB
Gatt descriptor added : status=0, srvc_handle=40, descr_handle=43
                   : descriptor UUID=00002902000010008000000805F9B34FB
Gatt characteristic added status : status=0, srvc_handle=40, char. handle=45
                   : characteristic UUID=00002AFE00001000800000805F9B34FB
Gatt descriptor added : status=0, srvc_handle=40, descr_handle=46
                   : descriptor UUID=00002902000010008000000805F9B34FB
Gatt service start : status=0, srvc_handle=40
c311

Enter a number between 0 to 1 to stop/start listen incoming connect, or -1 to cancel.

1

Enter a number between 0 to 1 to non-connect/connect advertising type, or -1 to cancel.

1
GATT is now prepare listening to role=Peripheral
Listen operation succeed

Enter the command to execute:
Wake BT controller right now!

GATT is now listening to role=Peripheral
Allow BT controller into sleep
```

If the operation is successful, the mobile phone can find "**Bluegate**" after opening BLEDeng. The search interface is shown in the following figure:

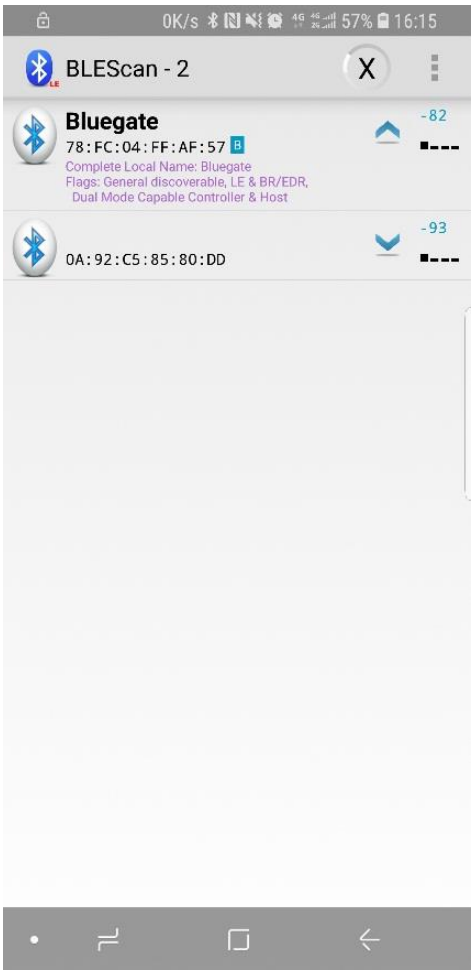


Figure 2: Find Bluegate

4.3. Add the Function of BLE Security Key

Enter the commands **c345** -> **0** -> **c334** –> **00002A3600001000800000805f9b34fb** -> **c336** -> **value of srv_handle** –> **00002A3600001000800000805f9b34fb** -> **2** -> **2** -> **c338** -> **1** -> **value of srv_handle** -> **c311** -> **1** -> **1**

In the commands, the value of srv_handle can be viewed after the first **00002A3600001000800000805f9b34fb** through the printed log. Please see an example as follows:

```
/data # ./demo_UI
Register CBK success


************************************************************
****** nFore BlueGate Bluetooth Middleware (V0.3.2.75) *****
************************************************************

Enable BT stack success

Enter the command to execute:
BT Adapter must reset hardware.

Get BT Userial fd: 16

Local device infos:
    Name:   [Bluegate]
    Addr:   [A4:04:50:30:26:31]
    Device Sleep


Wake BT controller right now!

bt state: enable

Local device infos:
    Name:   [Bluegate]
    Addr:   [A4:04:50:30:26:31]
    Device Connectable


Client Register status=0
Service Register status=0
Allow BT controller into sleep

c345

Set GATT role to Peripheral(0), Central(1), or -1 to cancel.

0
Gatt set role : status=0, role=0

Enter the command to execute:
c334

Enter the service uuid as format 00001101000010008000000805f9b34fb or -1 to cancel.

00002A3600001000800000805f9b34fb
```

```
Enter the command to execute:
Gatt service added : status=0 srvc_handle=40
                   : service UUID=00002A3600001000800000805F9B34FB
c336

Enter service handle number, or -1 to cancel.

40

Enter the characteristic uuid as format 00001101000010008000000805f9b34fb or -1 to cancel.

00002A3600001000800000805f9b34fb

Enter characteristic properties, or -1 to cancel.

2

Enter characteristic permit, or -1 to cancel.

2

Enter the command to execute:
Gatt characteristic added status : status=0, srvc_handle=40, char. handle=42
                                 : characteristic UUID=00002A3600001000800000805F9B34FB
c338

Enter a number to stop(0) or start(1) service, or -1 to cancel.

1

Enter a number a service handle, or -1 to cancel.

40

Enter the command to execute:
Gatt service start : status=0, srvc_handle=40
c311

Enter a number between 0 to 1 to stop/start listen incoming connect, or -1 to cancel.

1
```

```
Enter a number between 0 to 1 to non-connect/connect advertising type, or -1 to cancel.

1
GATT is now prepare listening to role=Peripheral
Listen operation succeed

Enter the command to execute:
Wake BT controller right now!

GATT is now listening to role=Peripheral
Allow BT controller into sleep

Wake BT controller right now!

GATT is now connected to addr: [49:79:87:47:31:ED], role=Peripheral
Allow BT controller into sleep

Wake BT controller right now!
```

After the operation is completed, please open BLEDeng on the phone, click to enter Bluegate, click **Intermediate Cuff Pressure**, then click **R**, and then enter **y** -> **7127715** in sequence in the console according to the Bluetooth pairing request. After the phone is paired successfully, the mobile phone can receive the data sent by the module.

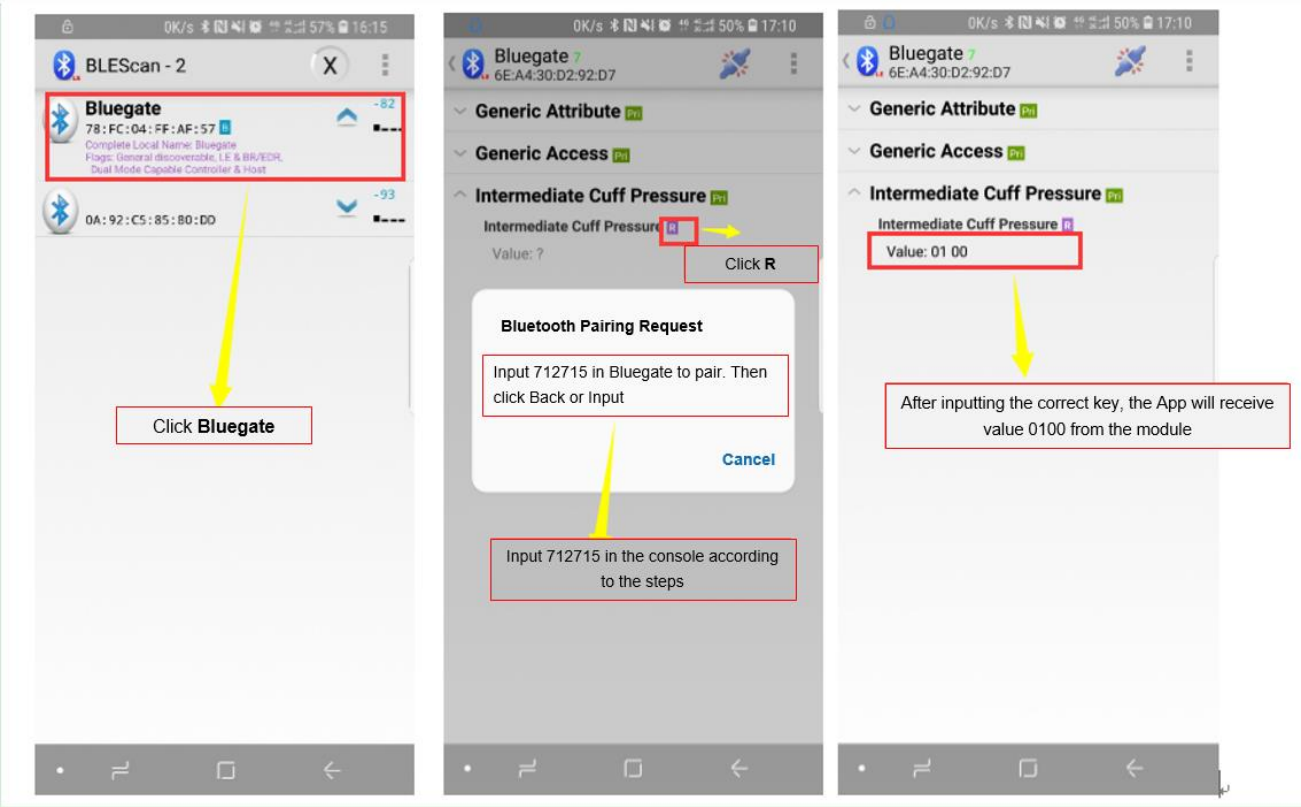
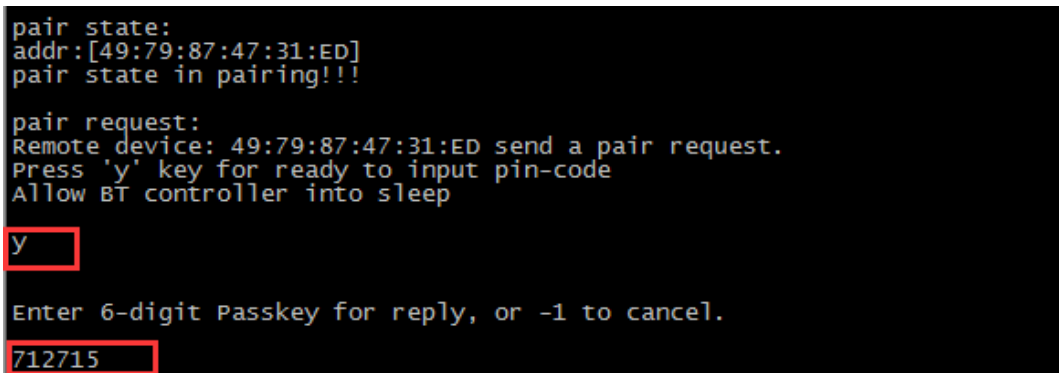


Figure 3: Add Security Key

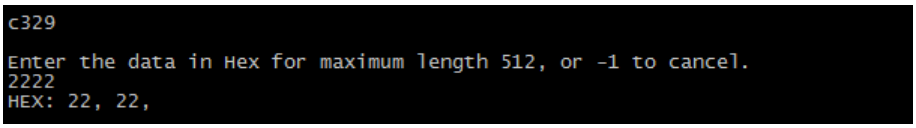


4.4. Data Transmission

4.4.1. Send Data from Module to Mobile Phone

Firstly, enter the commands **c345** -> **0** -> **c328** -> **c311** -> **1** -> **1** in sequence to enable the BLE function. After that, the mobile phone can find the broadcast Bluegate. Click to enter **Bluegate**, and click "**18FF**", then a radio frequency icon will be shown on the right. Click the icon to see if it turns to be solid; if it is solid, it means that the Bluetooth connection is established. After that, data transmission can be performed, such as sending data "**2222**". The example is as follows:

```
c329
2222
```



After the data transmission is completed, the data "**2222**" sent by the module can be received in the application of the mobile phone, as shown in the following figure:

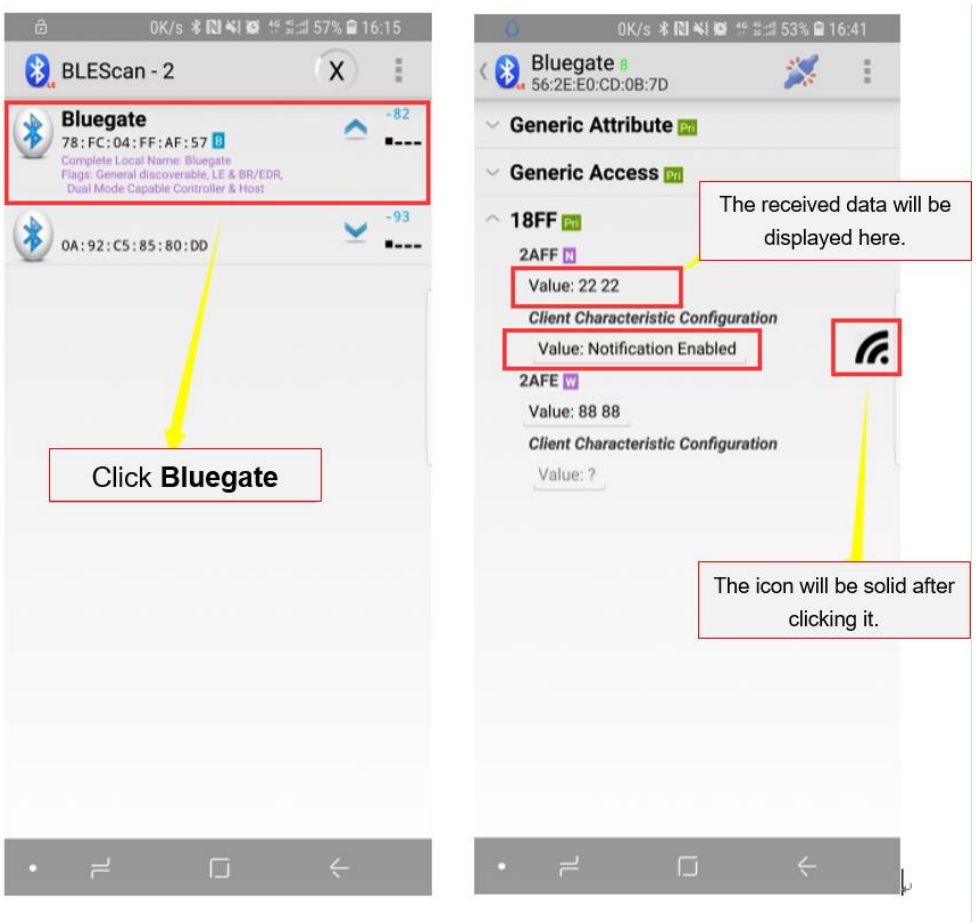


Figure 4: Send "2222" from Module to Mobile Phone

4.4.2. Send Data from Mobile Phone to Module

Firstly, enter the commands **c345** -> **0** -> **c328** -> **c311** -> **1** -> **1** in sequence to enable the BLE function. After that, the mobile phone can find the broadcast **Bluegate**. Click to enter **Bluegate**, and click "**18FF**". After that, input the value to be sent to the module in the "Value" of 2AFE such as inputting "**8888**". After sending the data, the received data can be viewed in the log of the console. The example is as follows:

```
Gatt Request write event :  [5F:3D:88:BC:85:2D], trans_id=1, attr_handle=45, offset=0, is_prep=0, length=2, need_rsp=1
                : data=8888
Wake BT controller right now!

Allow BT controller into sleep
```

4.5. Music Playback and Call Recording

Before implementing the functions of music playback and call recording, Bluetooth pairing is required, the steps are as follows:

Step 1: Enter the command **amix 'SEC_AUX_PCM_RX Audio Mixer MultiMedia1' 1** on the command line to run demo_UI, and enter **help** to view the operation commands.
Step 2: Enter **pair** or **c9**.
Step 3: Enter the Bluetooth MAC address of the phone. Click "**Confirm**" on the mobile phone, and "**y**" needs to be input on the console to successfully pair.

```
****************************************** Help ******************************************
bt_enable(c1)           : Enable BT stack.
bt_disable(c2)          : Disable BT stack.
localinfos(c3)          : Get local device infos.
setname(c4)             : Set local device name.
setscan(c5)             : Set scan mode.
scan(c6)                : Scan nearby bluetooth devices.
stop(c7)                : Stop scan nearby bluetooth devices.
pairlist(c8)            : List paired remote device.
pair(c9)                : Pair with remote device.
delete(c10)             : Delete paired remote device.
connect(c13)            : Connect to paired remote device.
disconnect(c14)         : Disconnect to paired remote device.
switch(c15)             : Switching current remote device.
setprofile(c16)         : set enable profile
getprofile(c17)         : get enable profile
address(c18)            : get bluetooth address of local device
****************************************** HFP ******************************************
```

4.5.1. Music Playback

After Bluetooth pairing is successful, please plug in Codec and earphone in Quectel EVB, and switch to CODEC mode. After the mobile phone plays music, the headset can hear the music played on the mobile phone. The switch for CODEC mode is shown below:

Figure 5: Switch for CODEC Mode

4.5.2. Call Recording

Step 1: Please switch the Quectel EVB to BT mode. Before running demo_UI, please enter the following recording command on the command line:

```
# amix 'MultiMedia1 Mixer AUX_PCM_UL_TX' 1
# arec –C 1 –R 8000 /data/record.wav &
# /data/demo_UI
```

Step 2: Perform Bluetooth pairing. Make a call by the mobile phone, and the downstream voice will be

recorded. After hanging up the phone, execute the following command to play the call recording */data/record.wav*:

```
# aplay /data/record.wav
```

NOTE

Because the recording is before running demo_UI, there is no sound at the beginning of the recorded audio.

4.6. Wake up from Sleep

Before testing the wake-up function, please insert the Bluetooth antenna for good signal quality.

4.6.1. Wake up the Module by NF3303

When the AutoSleep function is enabled, the module enters the sleep mode, and the serial port is not able to perform data interaction. For AutoSleep function, please refer to ***document [4]***.

At this time, the mobile phone can still scan NF3303. When the mobile phone initiates the connection with NF3303, NF3303 pulls up BT_HOST_WAKE_UP to generate an interrupt and wake up the module. At this time, the application holds the walelock. Once the data interaction is completed, NF3303 pulls down BT_HOST_WAKE_UP to generate a falling edge interrupt; thus the wakelock is released. If no other task holds the wakelock, the module enters sleep mode, and its serial port cannot perform any data interaction.

4.6.2. Wake up NF3303 by the Module

When there is no data interaction, the Bluetooth protocol stack outputs callback function notifications. After the application receives a notification, it pulls down the BT_HOST_WAKE_UP pin, allowing the NF3303 to sleep. When the module needs to transmit data with NF3303, the Bluetooth protocol stack outputs a callback function notification. The application pulls up the BT_HOST_WAKE_UP pin to wake up NF3303 after it receives the notification. An oscilloscope can be used to monitor the level change of the BT_DEVWAKE pin in real time, test the current consumption of the NF3303, and verify whether NF3303 sleeps and wakes up normally.

5 Appendix A References

Table 2: Related Documents

SN	Document Name	Description
[1]	nFore Bluegate Programming Guide 0.10.4	BSA protocol stack API programming guide
[2]	Quectel_EC20_R2.1-QuecOpen_Hardware_Design	Quectel EC20 R2.1 QuecOpen hardware design
[3]	SA-HRD-211 NF3303 Module Hardware Specification V1.1	NF3303 WIFI/BT module specification
[4]	LTE_Standard_Series_QuecOpen_Low_Power_Mode_Application_Note	LTE standard series QuecOpen low power mode application note

Table 3: Terms and Abbreviations

Abbreviation	Description
BLE	Bluetooth Low Energy
BLSP	BAM Low-Speed Peripherals
CTS	Clear To Send
GPIO	General-purpose input/output
HCI	Host Controller Interface
I2C	Inter-Integrated Circuit
RTS	Require To Send
SDK	Software Development Kit
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver & Transmitter