FSC-BT3431 User Guide

Release 3.5.1

Table of contents

1 Hardware Design **2**

1.1 Module Pin Diagram . . . 2

1.2 Pin Description . . . 3

1.3 Hardware Design Note . . . 3

2 Functional Description **5**

2.1 Default Configuration . . . 5

2.2 GPIO Indications . . . 5

2.2.1 LED . . . 5

2.2.2 BT Connection Status . . . 5

2.3 GATT Service . . . 6

2.4 Working Mode . . . 6

2.4.1 Throughput Mode . . . 6

2.4.2 Command Mode . . . 6

2.5 Low-Power Strategy . . . 6

2.5.1 Low-Power Mode Setting . . . 6

2.5.2 Low-Power Operation Strategy . . . 6

2.6 Data Rate (Typical) . . . 7

3 Data Communication Principles **8**

3.1 Working Principle . . . 8

3.2 MCU-to-Module Communication . . . 9

3.3 Module-to-Module Communication . . . 10

3.4 Module-to-Phone Communication . . . 10

3.4.1 Why is an APP required on a mobile phone for Bluetooth connection and communication? . . . 10

3.4.2 Communication Application . . . 11

4 Quick Development Kit **12**

4.1 Datasheet . . . 12

4.2 AT Command Set . . . 12
4.3 Serial Port Tool . . . 12
4.4 App&SDK . . . 12
4.5 Firmware Upgrade . . . 12
4.5.1 OTA Upgrade . . . 12

5 Quick Start **14**
5.1 What you need . . . 14
5.1.1 Required Hardware . . . 14
5.1.2 Software and Setup . . . 14
5.2 Hardware Access . . . 14
5.2.1 UART Connection . . . 14
5.3 Communication Test . . . 15
5.3.1 Module-to-PC . . . 15
5.3.2 UART Communication Test . . . 17

6 Development Examples **18**
6.1 Data Throughput Mode Application . . . 18
6.1.1 What is Throughput Mode? . . . 18
6.1.2 Module to Phone Application . . . 19
6.1.3 Module to Module Application . . . 19
6.2 Read/Write Module Default Parameters . . . 21
6.3 Data Transmission Flow . . . 22
6.4 Module Acts as Master to Connect to Remote Device . . . 23

7 Firmware Upgrade **25**
7.1 OTA Upgrade . . . 25

8 FAQs **27**
8.1 Why is an APP required on a mobile phone for Bluetooth connection and communication? . . . 27
8.2 How to obtain Bluetooth MAC address on iOS device? . . . 27

9 Contact Information **30**

10 Appendix **31**

[中文版]

This guide is applicable to:

FSC-BT3431 Bluetooth BLE data throughput transmission module, which supports the BLE (GATT Client/GATT Server) Bluetooth protocol.

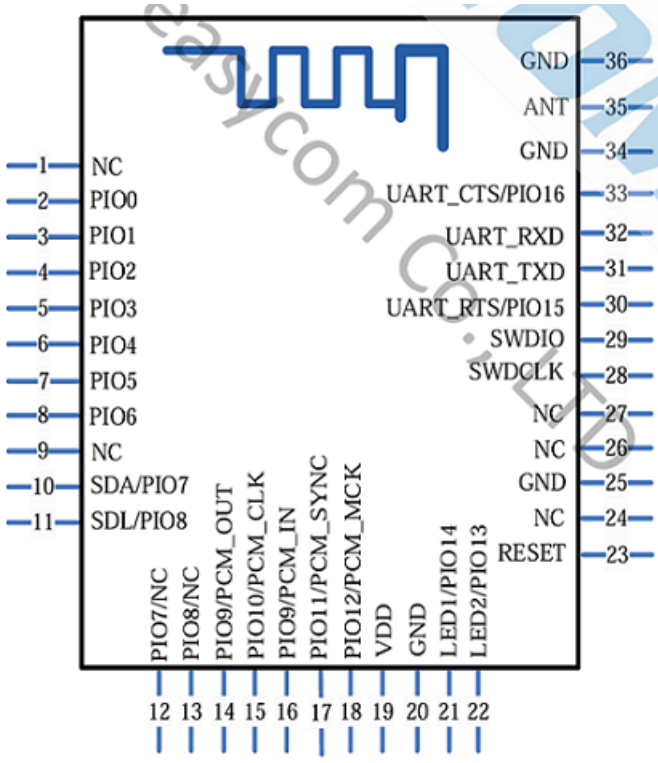
This guide consists of the following parts:

Chapter 1

Hardware Design

[中文版]

1.1 Module Pin Diagram



FSC-BT3431 PIN Diagram (Top View)

1.2 Pin Description

Pin	Pin Nam	Type	Pin Descriptions
31	UAR	O	UART data pin
32	UAR	I	UART data pin
23	RE-SET	I	Active-low reset
19	VDD	Powe	3.3V power supply.LDO (Low Dropout Regulator) power supply is recommended
20	GND	GND	GND
28	SWC	I/O	Programming pin
29	SWD	I/O	Programming pin
8	SLP_	O	After Bluetooth connection, the module pulls low to notify the customer’s MCU to exit sleep mode, and outputs serial data with a 10ms delay.After Bluetooth disconnection, the module pulls high to notify the customer’s MCU to enter sleep mode.
7	WAK	I	When the MCU pulls high, the module enters sleep mode.when the MCU pulls low, the module exits sleep mode
21	LED	O	Outputs a square wave when Bluetooth is disconnected.outputs high level when Bluetooth is connected
22	STA-TUS	O	Low level : Bluetooth is disconnected.High level : Bluetooth is connected.

1.3 Hardware Design Note

- The module can be used by simply connecting VDD/GND/STATUS/UART_RX/UART_TX.
- If the MCU needs to obtain the connection status of the Bluetooth module, the STATUS pin (Pin 22) must be connected.
- If the user needs to check the working status of the module, the LED pin (Pin 21) must be connected.
- If low-power consumption is required, connect Pin 8 (SLP_IND) and Pin 7 (WAKE_UP); leave these pins floating if low-power consumption is not needed.

- After completing the schematic diagram, please send it to Feasycom for review to ensure the Bluetooth communication distance reaches the optimal performance.

Chapter 2

Functional Description

[中文版]

2.1 Default Configuration

Name	FSC-BT3431
UART Baudrate	115200/8/N/1

2.2 GPIO Indications

2.2.1 LED

PIN	Status	Description
PIN 21	1Hz Square Wave	Bluetooth Disconnected
PIN 21	High Level	Bluetooth Connected

2.2.2 BT Connection Status

PIN	Status	Description
PIN 22	Low Level	Bluetooth Disconnected
PIN 22	High Level	Bluetooth Connected

2.3 GATT Service

Type	UUID	Permission	Description
Service	0xFFF0		Throughput transmission service
Write	0xFFF2	Write， Write Without Response	Sent from APP to Module
Notify	0xFFF1	Notify	Sent from Module to APP

2.4 Working Mode

2.4.1 Throughput Mode

- **Bluetooth Not Connected**: Data received via UART is parsed as AT commands.
- **Bluetooth Connected**: All data received via UART is sent as-is to the remote Bluetooth device.

2.4.2 Command Mode

- **Bluetooth Not Connected**: Data received via UART is parsed as AT commands.
- **Bluetooth Connected**: Data received via UART is still parsed as AT commands. Data must be sent to the remote device using AT commands, e.g., `AT+LESEND`.

2.5 Low-Power Strategy

2.5.1 Low-Power Mode Setting

The low-power function can be enabled or disabled via the command `AT+LPM{=Param}`.

2.5.2 Low-Power Operation Strategy

- **AT+LPM=1**
  - BT automatically enters low-power mode if no data is received from the BT UART for more than 5 second.
  - The first frame of data from the BT UART wakes up Bluetooth.
  - Does not enter low-power mode after a successful connection.

- **AT+LPM=2**
  - When the MCU pulls PIN7 high, BT enters sleep mode.
  - when the MCU pulls PIN7 low, BT exits sleep mode.
  - After Bluetooth connection, BT pulls PIN8 low to notify the MCU to exit sleep mode, and outputs data with a 10ms delay.
  - After Bluetooth disconnection, BT pulls PIN8 high to notify the MCU to enter sleep mode.

2.6 Data Rate (Typical)

Bau-drate	Data Packet	Transmission Interval	Connection Interval	Transmission Method	Rate
921600	244	20ms	7.5ms	Notify	12000 Byte/s

Chapter 3

Data Communication Principles

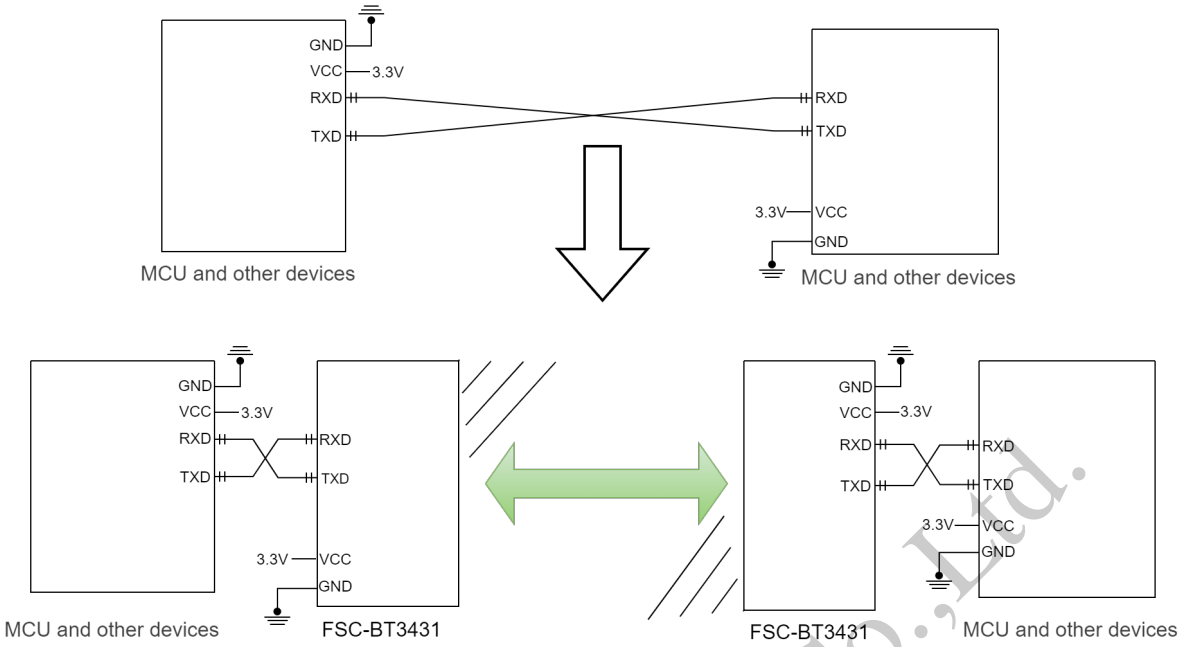
[中文版]

3.1 Working Principle

FSC-BT3431 Bluetooth BLE data transmission modules enable wireless communication between devices based on the BLE (Bluetooth Low Energy) protocol.

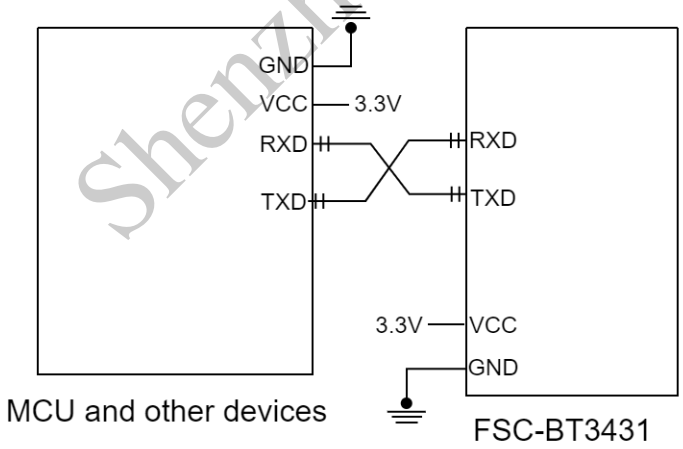
- **BLE** : It adopts an event-driven low-power architecture and defines a “Service-Characteristic” model through the GATT protocol to realize intermittent small-data interaction (e.g., sensor data), making it suitable for IoT (Internet of Things) devices.

The module sends AT commands or transparent transmission data to the host device (mobile phone/MCU) via UART to complete connection establishment, data exchange, and status management.



As shown in the diagram, the Bluetooth module is used to replace the physical wires in full-duplex communication. A device such as a microcontroller unit (MCU) (on the left) sends data to the left-side Bluetooth module via its TXD pin. After the RXD pin of the left-side Bluetooth module receives the serial data, it automatically transmits the data via radio waves to the remote Bluetooth module (on the right). The right-side remote Bluetooth module then receives the over-the-air data and sends it to the local device (e.g., an MCU) on the right via its TXD pin.

3.2 MCU-to-Module Communication



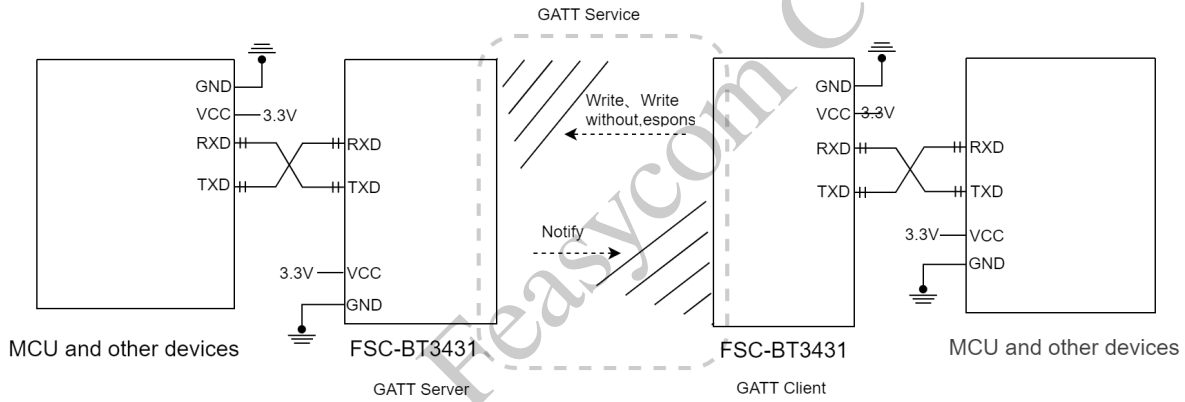
This diagram shows a connection schematic of a main control MCU (Microcontroller Unit) and an FSC-BT3431 Bluetooth module. Command interaction between the main control and the Bluetooth module is realized through serial cross-connection, supporting wireless communica-

tion functions, which is applicable to scenarios such as IoT devices and remote control.

1. **Serial Communication Interface** : The transmitting end of the main MCU (MCU_TX) is cross-connected with the receiving end of the Bluetooth module (UART_RX), and the receiving end (MCU_RX) is similarly connected to the transmitting end of the Bluetooth module (UART_TX), forming a two-way data transmission channel.
2. **Power Supply and GND** : The Bluetooth module is connected to 3.3V via the VDD_3V3 pin and shares a common GND with the main MCU to ensure level compatibility and signal stability.

3.3 Module-to-Module Communication

Two FSC-BT3431 Bluetooth modules can establish a Bluetooth connection when powered on.



FSC-BT3431 module has master-slave device functions; the left module can be configured as a master device, and the right module as a slave device. The master device can send commands to realize Bluetooth scanning, connection establishment, data transmission, disconnection, etc. Among them, the device that actively initiates a Bluetooth connection is defined as a master device, and the device that receives a connection request is a slave device.

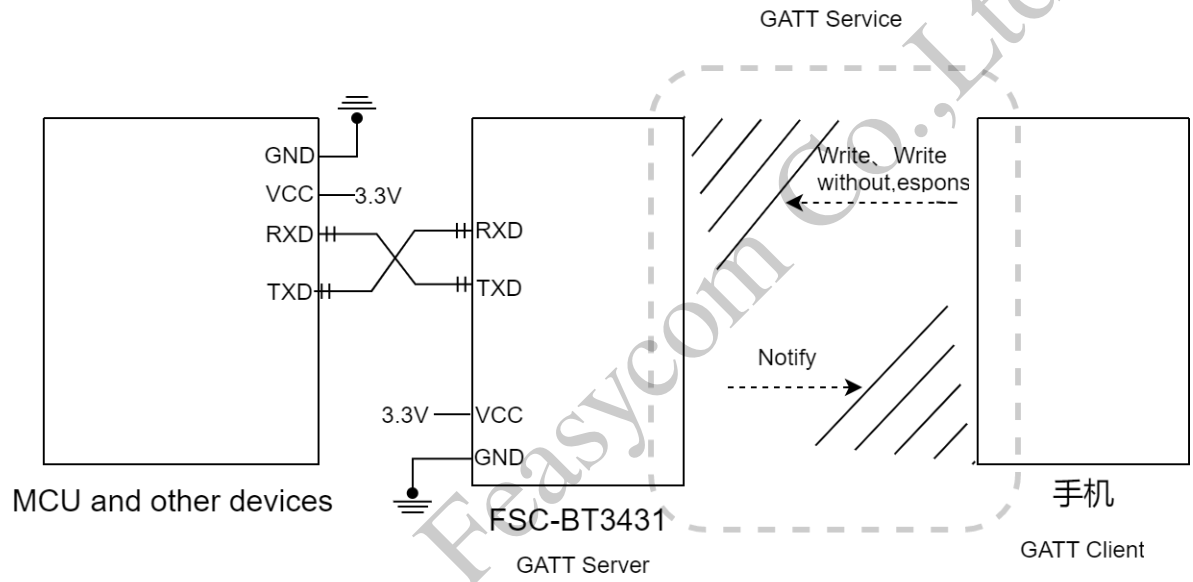
3.4 Module-to-Phone Communication

3.4.1 Why is an APP required on a mobile phone for Bluetooth connection and communication?

The native Bluetooth function of mobile phones only supports general scenarios, such as audio transmission and file transfer. Some Bluetooth peripheral devices can be connected through the mobile phone's

built-in settings program, such as Bluetooth speakers, Bluetooth headsets, Bluetooth keyboards, Bluetooth mice, etc. When a Bluetooth peripheral device cannot be connected by the mobile phone's native settings program (for example, the Bluetooth module only supports SPP/GATT protocols), to connect such a module, it is generally necessary to install a specific mobile application on the mobile phone, such as the FeasyBlue application

3.4.2 Communication Application



Bluetooth Module side (FSC-BT3431) : It will continuously send broadcast data outward when powered on;

Mobile Phone side : It can search and obtain the broadcast packet of the FSC-BT3431 module through the FeasyBlue App , and initiate a MAC address/UUID connection request to the module side (FSC-BT3431), while obtaining all services and characteristics provided by the device.

After a successful connection, the Bluetooth module (FSC-BT3431) will pull high the connection status pin and report the connection status command (valid in command mode) to notify the host side of the successful Bluetooth connection;

Host side：Data can be sent to the remote (mobile phone side) Bluetooth via the serial port through the Bluetooth module, and the remote (mobile phone side) Bluetooth can also send data to the host .

Chapter 4

Quick Development Kit

[中文版]

4.1 Datasheet

- FSC-BT3431 Datasheet

4.2 AT Command Set

- FSC-BT3431 General BLE Data AT Command Set

4.3 Serial Port Tool

- Feasycom Serial Port Tool：A serial communication analysis tool based on Windows PC.

4.4 App&SDK

- FeasyBlue：Feasycom App & SDK resource supporting Android and iOS, which enables functions such as Bluetooth BLE & SPP data communication test, Feasycom module firmware version reading, and parameter configuration and OTA AT commands etc.

4.5 Firmware Upgrade

4.5.1 OTA Upgrade

- Tool：NSUtil APP (A mobile application based on Android.)

- User Guide：Please refer to FSC-BT3431 - OTA Upgrade .

Chapter 5

Quick Start

[中文版]

5.1 What you need

5.1.1 Required Hardware

- 1 x FSC-BT3431 Module
- 1 x PC (Windows)

5.1.2 Software and Setup

- Feasycom Serial Port Tool：A serial communication analysis tool based on Windows PC.
- **Communication Interface**：UART
- **Serial Configuration**：115200/8/N/1 (General default for Feasycom firmware)

5.2 Hardware Access

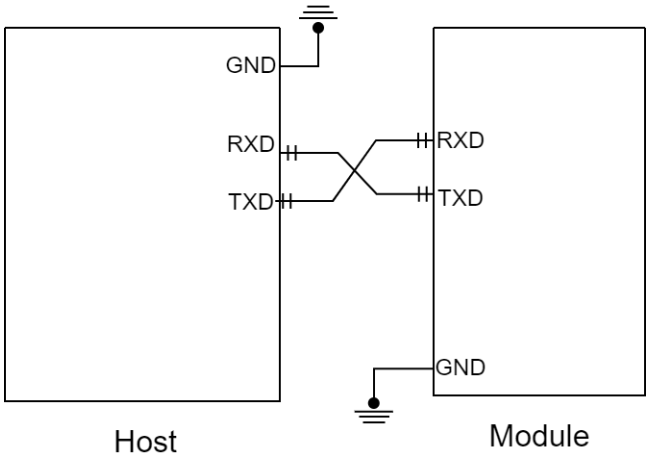
5.2.1 UART Connection

FSC-BT3431 provides one channels of Universal Asynchronous Receiver/Transmitters(UART)(Full-duplex asynchronous communications). The UART Controller performs a serial-to-parallel conversion on data received from the peripheral and a parallel-to-serial conversion on data transmitted from the CPU.

This is a standard UART interface for communicating with other serial devices. The UART interface provides a simple mechanism for communicating with other serial devices using the RS232 protocol.

When the module is connected to another digital device, UART_RX and UART_TX transfer data between the two devices.

When connecting the module to a host, please make sure to follow：

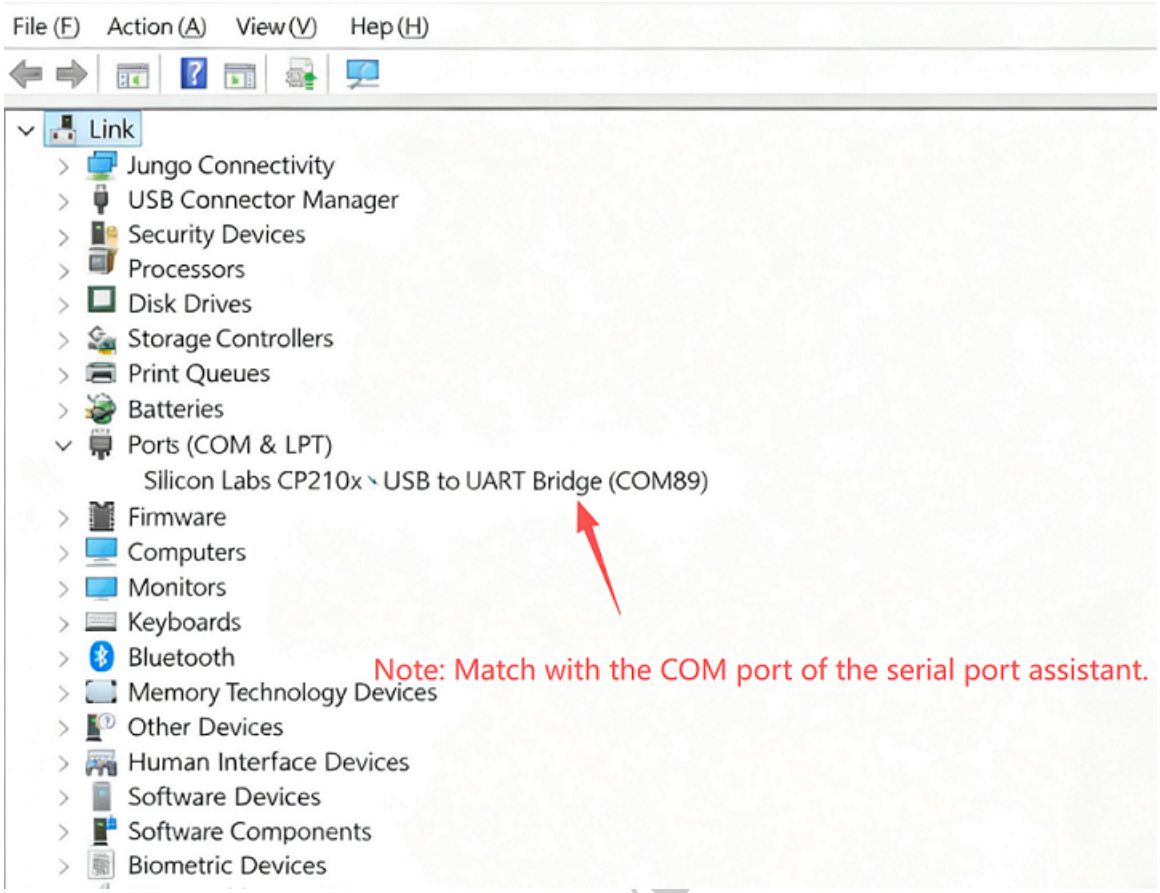


(UART Connection)

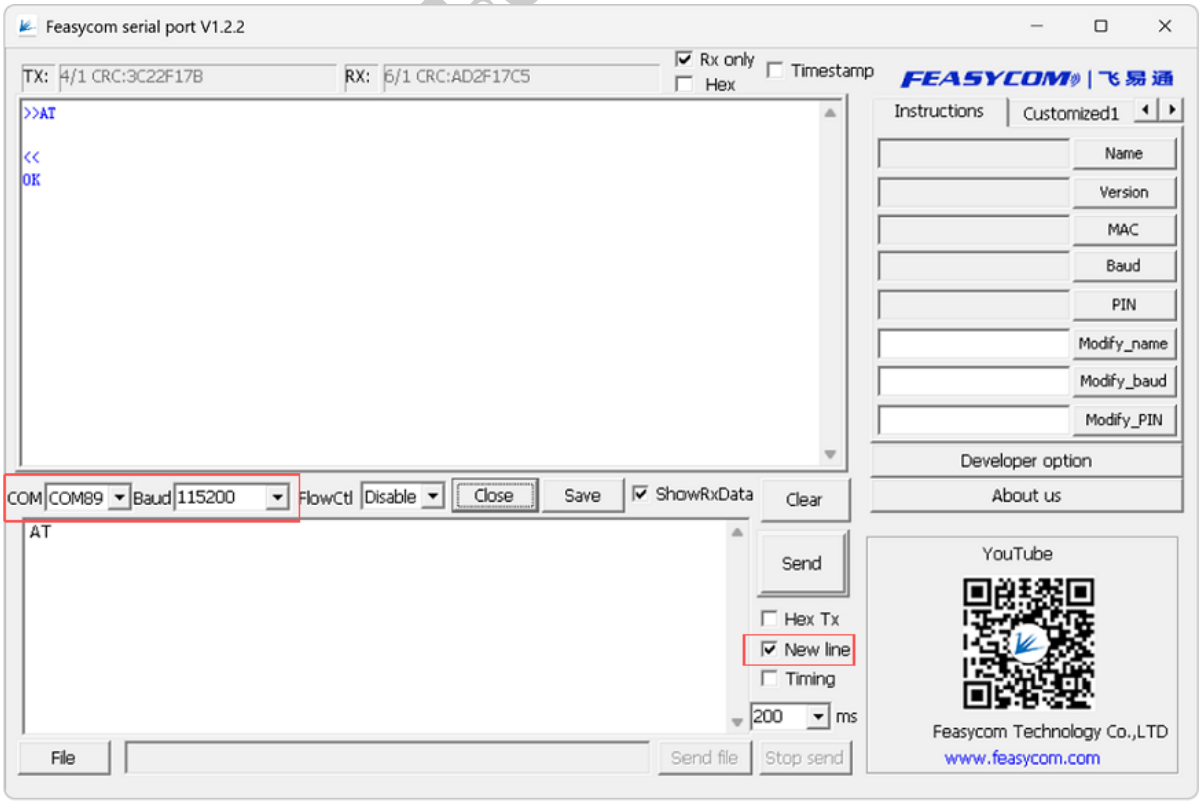
5.3 Communication Test

5.3.1 Module-to-PC

1.On the premise of ensuring compliance with the aforementioned **UART connection** requirements, set up the hardware testing environment with the PC. Upon completion, the PC will automatically recognize the device and virtualize a COM port.



2.Run the Feasycom Serial Port Tool on the PC, set the correct **COM** and **Baud**, and check the **New Line** opti.



5.3.2 UART Communication Test

The following lists a few basic general AT command test examples.

For more commands, please refer to FSC-BT3431 General BLE Data AT Command Set .

AT - UART Communication Test

Command	AT\r\n
Response	**\r\nOK\r\n**
Description	Test the UART communication between HOST and Module after power on, baudrate changed, etc.

Example:

```
send:     >>AT\r\n
response: <<\r\nOK\r\n    //Successfully connected.
```

AT+NAME - Read/Write Bluetooth Name

Example：Read Bluetooth name

```
send:     <<AT+NAME\r\n
response: >>\r\n+NAME=FSC-BT3431\r\n          //Default, please␣
↪refer to the actual reading result
response: >>\r\nOK\r\n
```

AT+VER - Read Current Firmware Version

Example：

```
Send:     <<AT+VER\r\n
Response: >>\r\n+VER=1.0.0,FSC-BT3431\r\n   //Example, please refer␣
↪to the actual reading result
response: >>\r\nOK\r\n
```

Chapter 6

Development Examples

[中文版]

6.1 Data Throughput Mode Application

6.1.1 What is Throughput Mode?

FSC-BT3431 Bluetooth BLE data module has two work modes: **Throughput Mode** and **Command Mode** .

The generic data throughput firmware for the FSC-BT3431 modules defaults to throughput mode. To switch modes, refer to FSC-BT3431 General Data AT Command Set and use *AT+TPMODE* command. The differences between the two work modes are as follows:

- **Throughput Mode** :

  Bluetooth Not Connected : Data received via UART is parsed as AT commands.

  Bluetooth Connected : All data received via UART is sent as-is to the remote Bluetooth device. It does not contain any data headers or framing and does not require AT commands to send data.

- **Command Mode** :

  Bluetooth Not Connected : Data received via UART is parsed as AT commands.

  Bluetooth Connected : Data received via UART is still parsed as AT commands. It will contain specific response indication headers and framing. Data

must be sent to the remote device using AT commands, such as *AT+LESEND*.

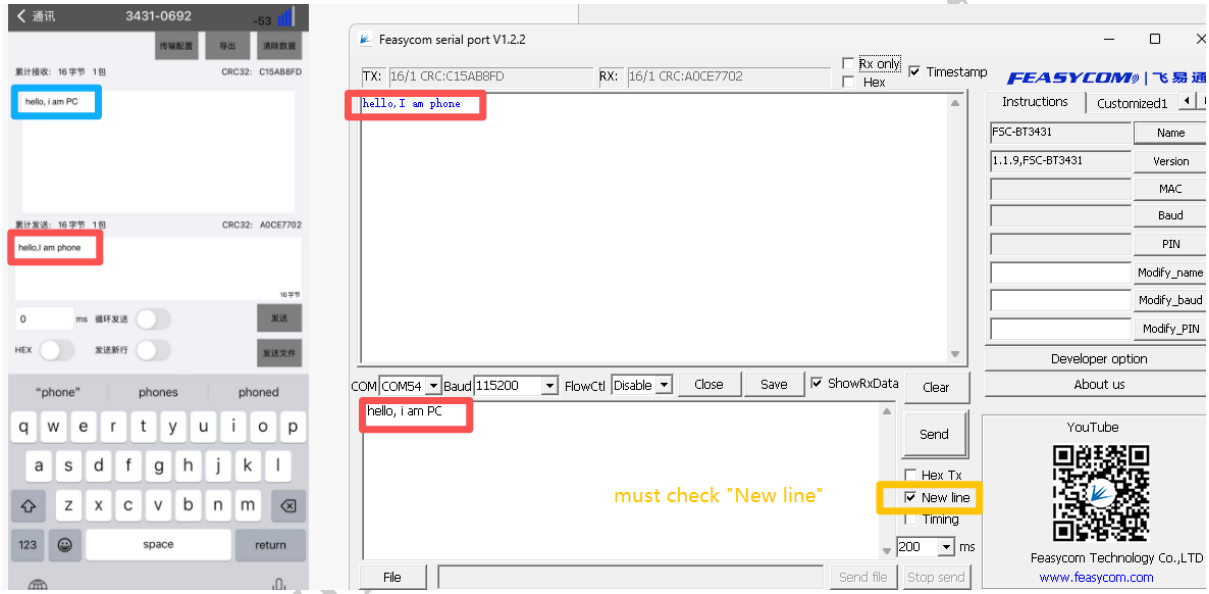
6.1.2 Module to Phone Application

1.Module Side: After power-on, the module will continuously send broadcast packet data;

2.Mobile Phone Side: Open the [FeasyBlue App] , scan for broadcast packets of nearby Bluetooth BLE devices, find the target Bluetooth module, and establish a connection;

3.After successful connection, the status pin of the module will pull up the level, indicating that the connection has been established;

4.After successful connection, in the transparent transmission mode, the module will automatically transmit the serial port data it receives to the remote end (mobile phone side) via air.



6.1.3 Module to Module Application

Demonstration of BLE communication data transparent transmission between FSC-BT3431(Module1) and FSC-BT3431(Module2) Bluetooth modules, as follows:

1.Scan for nearby BLE devices

FSC-BT3431 scans for nearby Bluetooth BLE devices.

```
1 Send: <<AT+SCAN=1                                                    //␣
  ↪Scan nearby Bluetooth BLE devices
2 Response: >>OK
3       >>+SCAN={                                                      //
```

(continues on next page)

(continued from previous page)

```
↪Scan Started
    >>+SCAN=0,0,DC0D30001882,-45,10,FSC-BT3431                    //
↪Module2
    >>+SCAN=1,1,DC0D3023CA1C,-71,14,0000-1007Z ggg
    >>+SCAN=2,0,DC0D30000015,-88,23,FSC-BT1038C-LE-AKM-0015
    >>+SCAN=3,0,DC0D300017A8,-85,11,FSC-BT3721V
    >>+SCAN=5,0,A4405B28A838,-89,10,AirGo AS01
    >>+SCAN=}                                                     //
↪Scan Completed
```

2.Establish BLE connection request

FSC-BT3431(Module1) establishe BLE protocol connection with FSC-BT3431(Module2) via the *AT+LECCONN* command as follows:

```
Send: <<AT+LECCONN=DC0D300018820      //Establish BLE connection␣
↪request
Response: >>OK
```

Warning

AT+LECCONN=Target Bluetooth MAC address + 1-bit address type.

How to obtain the address type:

Use AT+SCAN=1 for scanning. The second parameter in the return result is the address type, as in the following example:

```
//The address type is the second parameter, which is "0".
Response: >>+SCAN=0,0,DC0D30001882,-45,10,FSC-BT3431
```

3.BLE Connection Established Successfully

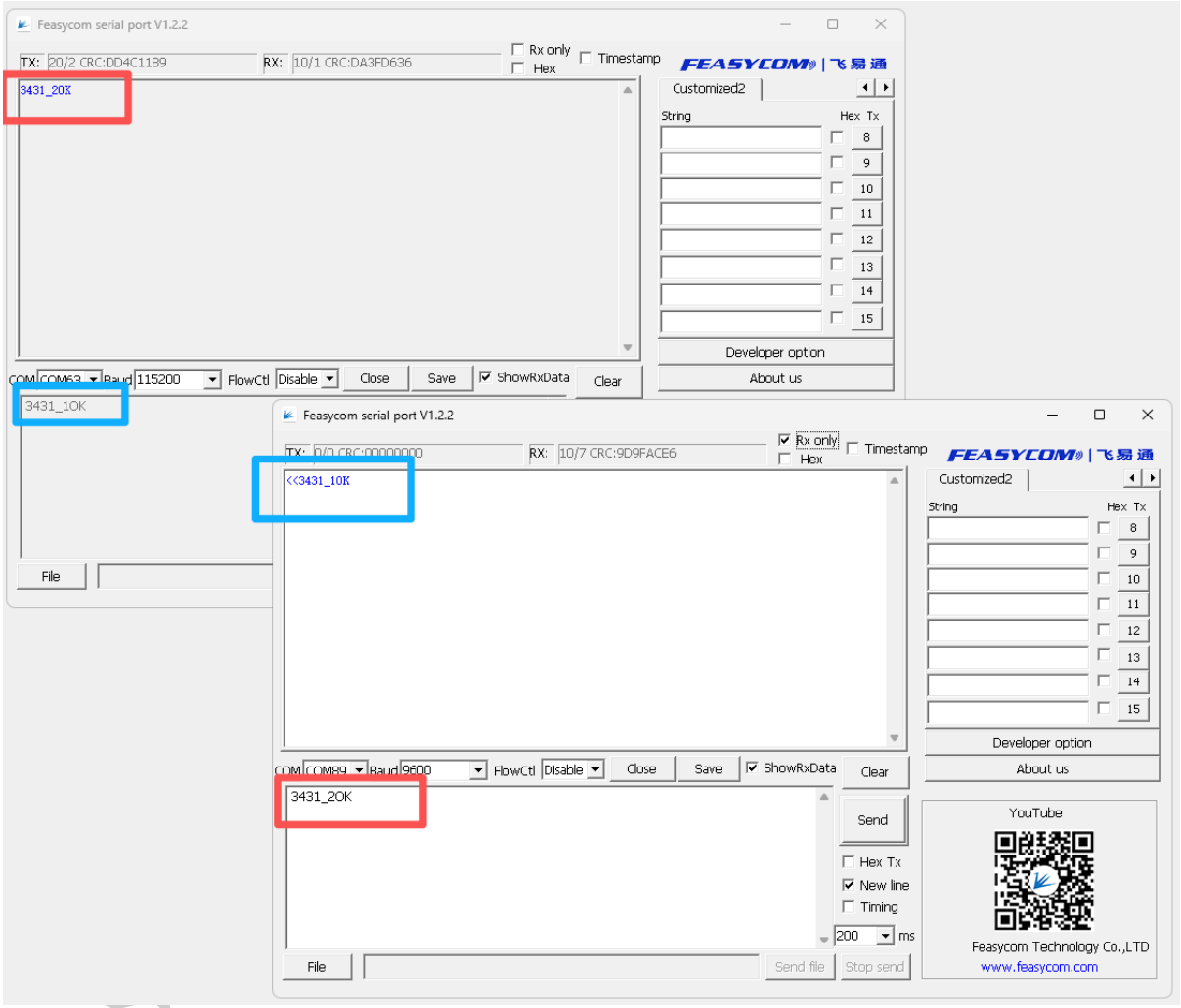
In throughput mode, after the Bluetooth connection is successfully established, the serial port cannot receive event response indicators. The current connection status can be determined by the level state of Pin22 (status indicator pin) of FSC-BT3431, as detailed below:

High Level (H): Indicates Bluetooth is successfully connected.

Low Level (L)： Indicates Bluetooth is not connected or the connection has been disconnected.

4.Send Data

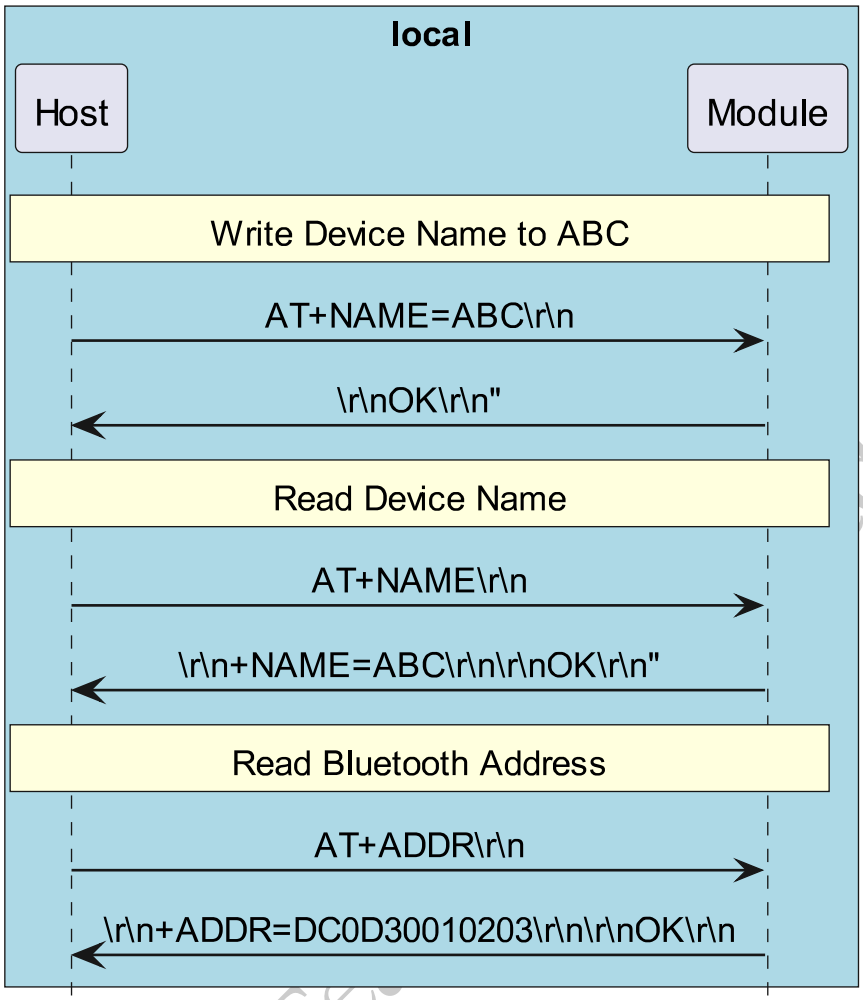
The throughput mode of the general data transmission firmware is enabled by default. After BLE connection is successfully established, data can be sent directly without the need to send data via AT commands:



6.2 Read/Write Module Default Parameters

When Bluetooth is not connected, the module parses UART data as AT commands. The host can query and modify the module’s default parameters. The following example demonstrates:

1. Write Device Name：ABC
2. Read Device Name
3. Read Bluetooth Address

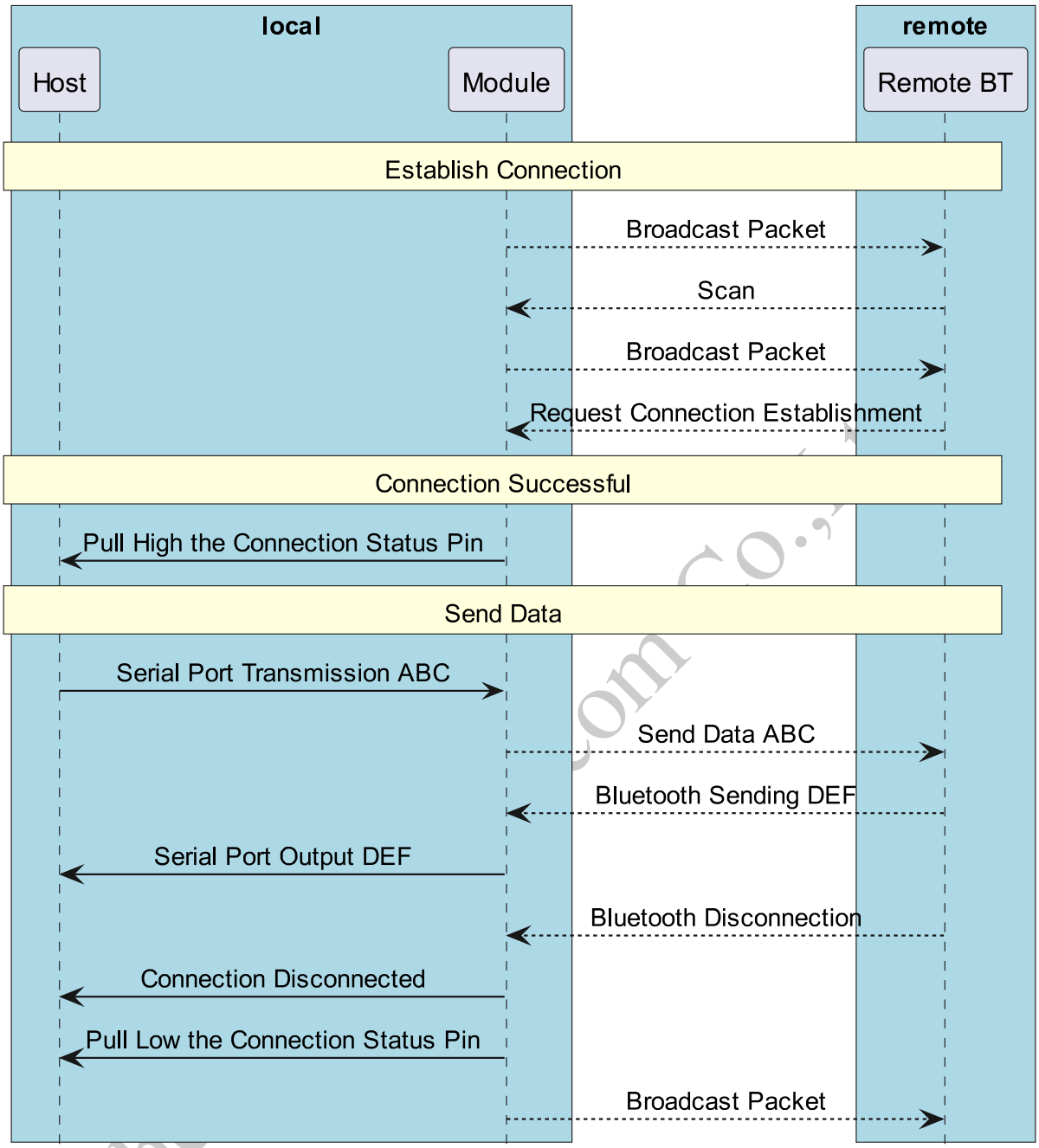


6.3 Data Transmission Flow

1.When the module is powered on, it continuously sends broadcast data outward. A remote Bluetooth device (e.g., a mobile phone) can obtain the broadcast packet by searching and initiate a connection request to the module.

2.After the connection is successfully established, the module will pull high the connection status pin to notify the host that the Bluetooth connection has been successfully established.

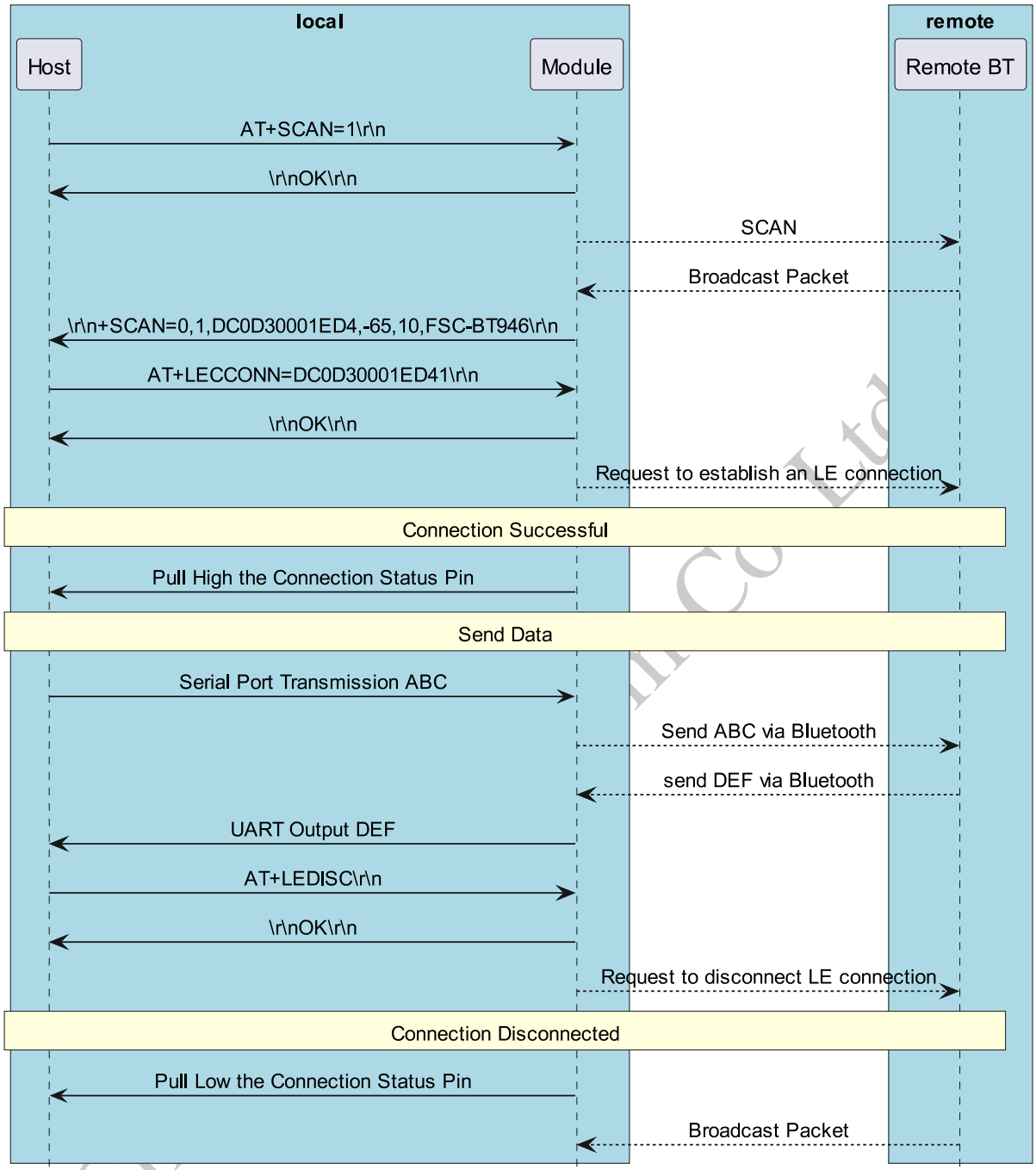
3.The host can send data to the remote Bluetooth device via the Bluetooth module, and the remote Bluetooth device can also send data to the host.



6.4 Module Acts as Master to Connect to Remote Device

The module can act as master device to connect to remote slave devices.

The host can send AT commands to control the module to perform scanning, connection, and disconnection operations. The following shows the process of connecting to other devices:


local
remote
Host
Module
Remote BT
AT+SCAN=1\r\n
\r\nOK\r\n
SCAN
Broadcast Packet
\r\n+SCAN=0,1,DC0D30001ED4,-65,10,FSC-BT946\r\n
AT+LECCONN=DC0D30001ED41\r\n
\r\nOK\r\n
Request to establish an LE connection
Connection Successful
Pull High the Connection Status Pin
Send Data
Serial Port Transmission ABC
Send ABC via Bluetooth
send DEF via Bluetooth
UART Output DEF
AT+LEDISC\r\n
\r\nOK\r\n
Request to disconnect LE connection
Connection Disconnected
Pull Low the Connection Status Pin
Broadcast Packet

Chapter 7

Firmware Upgrade

[中文版]

7.1 OTA Upgrade

1.Download and install the OTA upgrade application: NSUtil App (A mobile application based on Android).

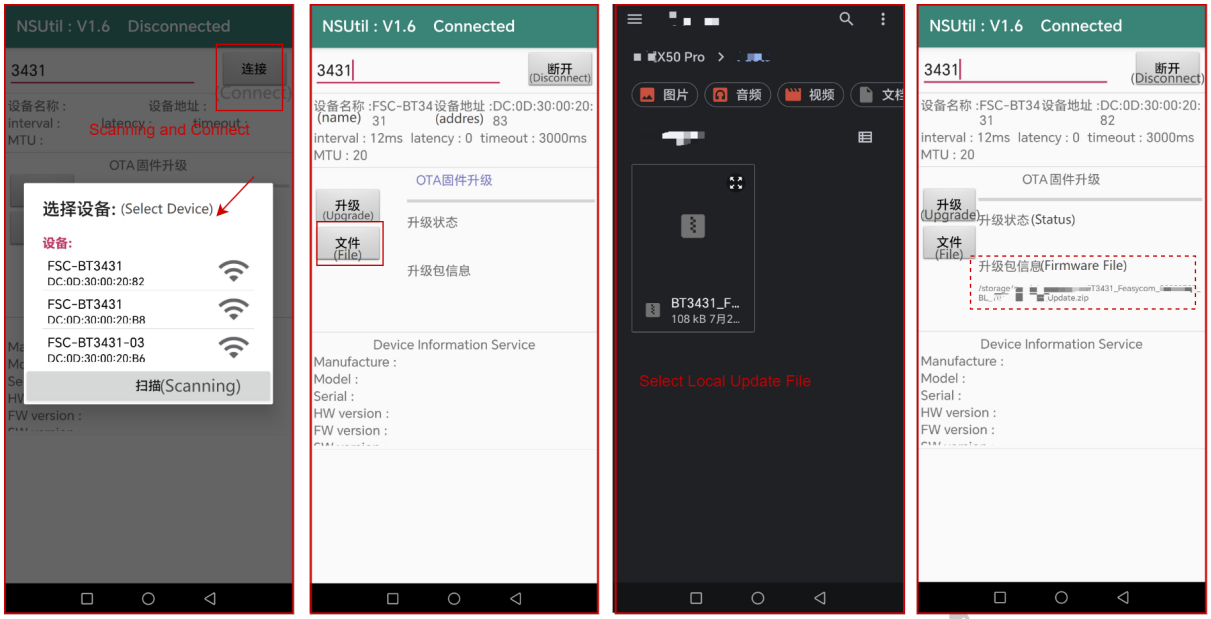
2.Scanning and connect : Open the **NSUtil** app, click the **Connect** to scan for and connect to the target Bluetooth device.

3.Load the firmware update file : Click the **File** button, select and import the firmware upgrade file stored locally on your mobile phone.

4.Enter upgrade mode : After successfully loading the firmware update file, click the **Upgrade** button to enter the upgrade mode. And the app will automatically reconnect to the Bluetooth module.

5.Start the upgrade : After the reconnection is successful, the upgrade will start automatically and the upgrade progress bar will be displayed.

6.OTA finish : Simply wait for the upgrade progress bar to finish and the **upgrade status** to display “**OTA finish**”.


NSUtil : V1.6 Disconnected
3431
连接
(Connect)
Scanning and Connect
选择设备: (Select Device)
设备:
FSC-BT3431
FSC-BT3431
FSC-BT3431-03
扫描(Scanning)
NSUtil : V1.6 Connected
3431
断开
(Disconnect)
interval : 12ms latency : 0 timeout : 3000ms
MTU : 20
OTA固件升级
升级
(Upgrade)
文件
(File)
升级状态
升级包信息
Device Information Service
Manufacture :
Model :
Serial :
HW version :
FW version :
X50 Pro
图片
音频
视频
BT3431_F...
108 kB 7月2...
Select Local Update File
NSUtil : V1.6 Connected
3431
断开
(Disconnect)
升级状态(Status)
升级包信息(Firmware File)
Update.zip
Device Information Service
Manufacture :
Model :
Serial :
HW version :
FW version :

(1) (2) (3) (4)


NSUtil : V1.6 Disconnected
3431
连接
(Connect)
设备名称 :
设备地址 :
interval :
latency :
timeout :
MTU :
OTA固件升级
升级
文件
升级状态(Status)
activate_new_image request...
升级包信息
等待蓝牙自动重连
(Wait for Bluetooth to auto-reconnect)
Device Information Service
Manufacture :
Model :
Serial :
HW version :
FW version :
NSUtil : V1.6 Connected
3431
断开
(Disconnect)
设备名称 : 设备地址 :DC:0D:30:00:20:84
interval : 50ms latency : 0 timeout : 5000ms
MTU : 244
OTA固件升级(OTA Upgrading)
升级
(Upgrade)
升级状态(Status)
文件
(File)
send_image_data...
升级包信息(Firmware File)
Device Information Service
Manufacture :
Model :
Serial :
HW version :
FW version :
NSUtil : V1.6 Disconnected
3431
连接
(Connect)
设备名称 :
设备地址 :
interval :
latency :
timeout :
MTU :
OTA固件升级(OTA Upgrading)
升级状态(Status)
OTA Finish in 30.263s, PRN : 2048
升级包信息
Device Information Service
Manufacture :
Model :
Serial :
HW version :
FW version :

(5) (6) (7)

Chapter 8

FAQs

[中文版]

8.1 Why is an APP required on a mobile phone for Bluetooth connection and communication?

The native Bluetooth function of mobile phones only supports general scenarios, such as audio transmission and file transmission. Some Bluetooth peripheral devices can be connected via the built-in settings program of the mobile phone, such as Bluetooth speakers, Bluetooth headsets, Bluetooth keyboards, Bluetooth mice, etc. When a Bluetooth peripheral device cannot be connected by the mobile phone's native settings program (for example, the Bluetooth module only supports SPP/GATT protocols), to connect such a module, it is generally necessary to install a specific mobile application on the mobile phone, such as the FeasyBlue application.

8.2 How to obtain Bluetooth MAC address on iOS device?

For security reasons, the iOS system converts the Bluetooth MAC address into a UUID at the underlying layer and sends it to upper-layer applications. Therefore, the APP cannot obtain the device’s MAC address.

FSC-BT3431 Bluetooth module will place the MAC address in the broadcast by default, and the APP can obtain the MAC address from the broadcast packet through the following methods.

```
- (void)centralManager:(CBCentralManager *)central␣
↪didDiscoverPeripheral:(CBPeripheral *)peripheral␣
↪advertisementData:(NSDictionary *)advertisementData RSSI:(NSNumber␣
↪*)RSSI
{
    if(![self describeDictonary:advertisementData])
    {
        NSLog(@"is not fsc module");
        return;
    }
}

- (Boolean)describeDictonary: (NSDictionary *) dict
{
    NSArray *keys;
    id key;
    keys = [dict allKeys];
    for(int i = 0; i < [keys count]; i++)
    {
        key = [keys objectAtIndex:i];
        if([key isEqualToString:@"kCBAdvDataManufacturerData"])
        {
            NSData *tempValue = [dict objectForKey:key];
            const Byte *tempByte = [tempValue bytes];
            if([tempValue length] == 6)
            {
                // tempByte 后面参数是蓝牙地址
                return true
            }
        }else if([key isEqualToString:@"kCBAdvDataLocalName"])
        {
            //there is name
            //NSString *szName = [dict objectForKey: key];
        }
    }
    return false;
```

(continues on next page)

(continued from previous page)

```
}
```

Chapter 9

Contact Information

Shenzhen Feasycom Co.,Ltd.

Address : Rm 508, Building, Fenghuang Zhigu, NO.50, Tiezai Road, Xixiang, Baoan Dist, Shenzhen, 518100, China.

Telephone : 86-755-23062695

Support : support@feasycom.com

Sales Service : sales@feasycom.com

Home Page : www.feasycom.com

Support Forum : forum.feasycom.com

Chapter 10

Appendix

PDF Download