FSC-BT910x User Guide

Release 3.5.0

Table of contents

[中文版]

This guide applies to the **FSC-BT910x** series Bluetooth dual-mode data transmission application modules, include:

- FSC-BT9101AI
- FSC-BT9104DI

This guide consists of the following parts:

Chapter 1

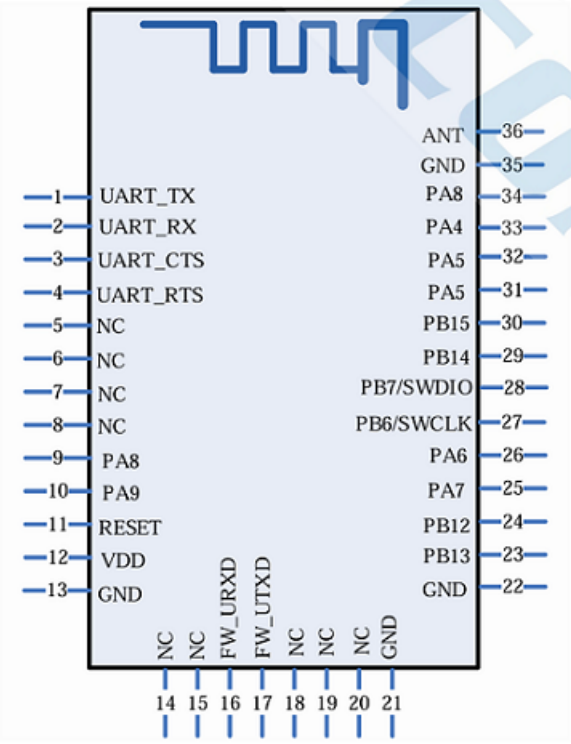
Hardware Design

[中文版]

1.1 Module Pin Description

1.1.1 FSC-BT9101AI

Module Pin Diagram:



Pin Description:

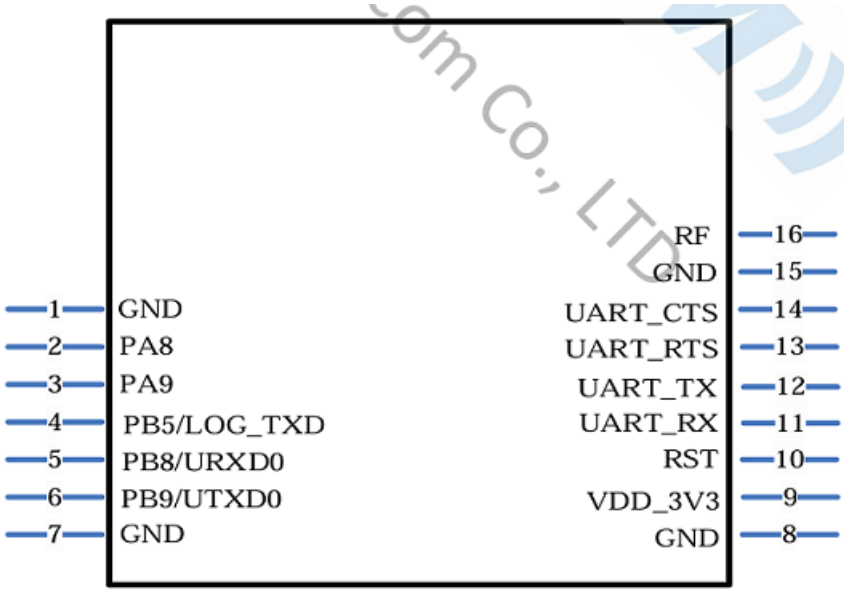
Pin	Pin Name	Type	Descriptions
1	UART_T	O	Serial Transmitter Data
2	UART_R	I	Serial Receiver Data
3	UART_C	I	UART Clear to Send
4	UART_R	O	UART Request to Send
9	Tran/CMI	I/O	UART mode control pin, H=command mode ,L=throughput mode
10	Disc	I/O	Disconnect the connecting pin
11	RESET	I	Active-low reset input
12	VDD	Powe	Power supply voltage 3.3V, LDO power supply preferred
13	GND	GND	GND
16	FW_URX	I	Programming Pin
17	FW_UTX	O	Programming Pin
32	Work	O	Bluetooth not connected to output square wave, Bluetooth connected to output high level
33	STA-TUS	O	Connection state, output. H=Connected , L=No connection
36	EXT_AN	ANT	Changing the 0 ohm resistance near the antenna, it is possible to connect a Bluetooth antenna externally

Note:

To use the PIN9 and PIN10 control modules AT+PIOCFG command is required to enable the function

1.1.2 FSC-BT9104DI

Module Pin Diagram:



Pin Description:

Pin	Pin Name	Type	Pin Descriptions
1	GND	GND	Power Ground
2	PA8	I/O	Programmable I/O
3	Status	O	Connection state, output. H=Connected , L=No connection
5	FW_URX	I	Programming Pin
6	FW_UTX	O	Programming Pin
8	GND	GND	GND
9	VDD_3V	VDD	Power supply voltage 3.3V, LDO power supply preferred
10	RESET	I	Active-low reset input
11	UART_R	I	Serial Receiver Data
12	UART_T	O	Serial Transmitter Data
13	UART_R	O	UART Request to Send
14	UART_C	I	UART Clear to Send
15	GND	GND	GND
16	EXT_AN	ANT	Changing the 0 ohm resistance near the antenna, it is possible to connect a Bluetooth antenna externally

1.2 Hardware Design Note

- In simple test environment, the module can be used for basic testing and operation by simply connecting **VDD**, **GND**, **UART_RX**, and **UART_TX**.
- If the MCU needs to obtain the connection status of the Bluetooth module, it needs to be connected to the STATUS pin.
- After drawing the schematic diagram, please send it to Feasycom for review to avoid Bluetooth distance not achieving the best effect.

Chapter 2

Functional Description

[中文版]

2.1 Default Configuration

Name	FSC-BT910x
LE-Name	FSC-BT910x-LE
Pin Code	0000
Secure Simple Pairing Mode	Off
UART Baudrate	115200/8/N/1

2.2 GPIO Indicators

2.2.1 LED

FSC-BT9101AI	Status	Description
Pin32	A square wave signal with a period of 1000ms, 200ms high level and 800ms low level	Bluetooth Disconnected
Pin32	High level	Bluetooth Connected

2.2.2 BT Connection Status

FSC-BT9101AI	FSC-BT9104DI	Status	Description
Pin33	Pin3	Low Level	Bluetooth Disconnected
Pin33	Pin3	High Level	Bluetooth Connected

2.3 Work Modes

2.3.1 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.

2.3.2 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+SPPSEND` or `AT+LESEND`.

2.4 GATT Service

Type	UUID	**Operation**	Description
Service	0xFFF0		Throughput transmission service
Write	0xFFF2	Write,Write Without Response	APP to Module
Notify	0xFFF1	Notify	Module to APP

Chapter 3

Data Communication Principles

[中文版]

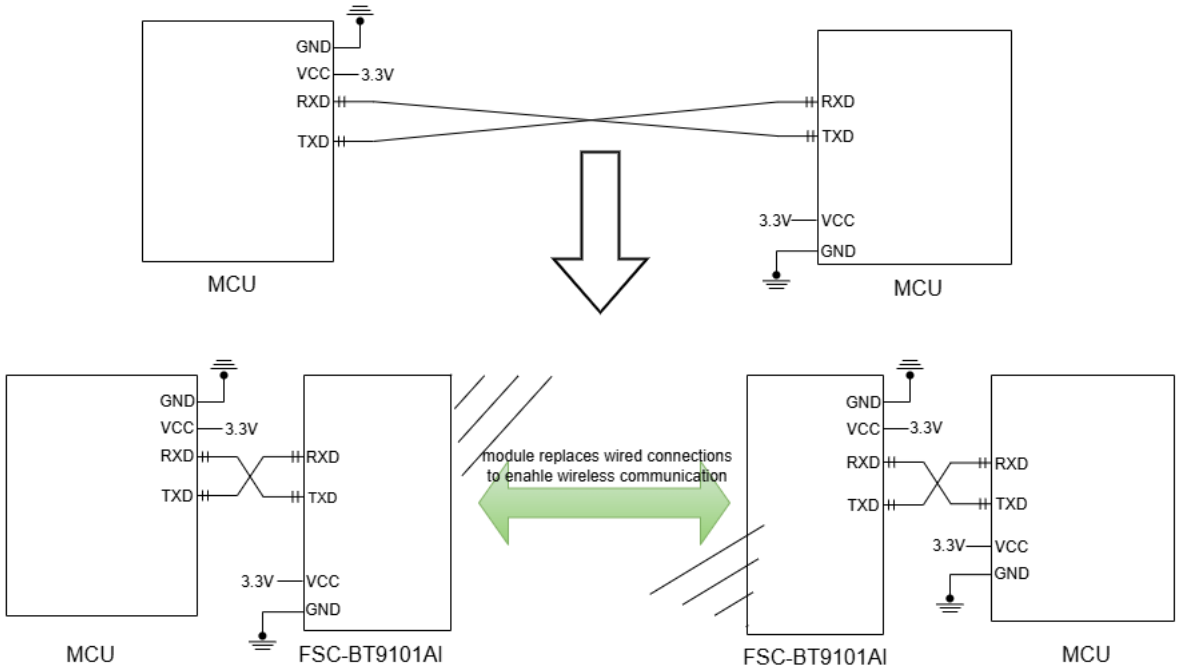
Taking the FSC-BT9101AI module as an example:

3.1 Communication Principle

FSC-BT910x series Bluetooth dual-mode data modules enable wireless communication between devices based on the SPP (Serial Port Profile) and BLE (Bluetooth Low Energy) dual-mode protocols.

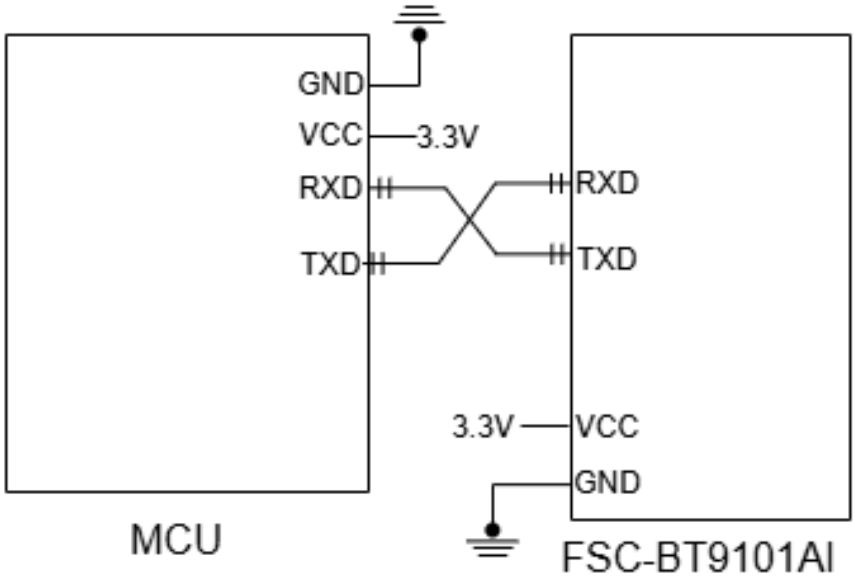
- **SPP Mode**: Emulates traditional serial port communication, establishing a virtual serial link through the RF layer. It supports continuous large data transfers (e.g., file transfer) and is suitable for scenarios like printers.
- **BLE Mode**: Utilizes an event-driven, low-power architecture, defining a “Service-Characteristic” model via the GATT protocol for intermittent small data interactions (e.g., sensor data), ideal for IoT devices.

Both modes share the underlying RF hardware and switch automatically via the protocol stack. The module communicates with the host device (phone/MCU) through the UART interface using AT commands or transparent data transmission to establish connections, exchange data, and manage status.



As shown in the figure, the Bluetooth module replaces the physical cable in full-duplex communication. A device like an MCU (left) sends data via its TXD pin to the Bluetooth module (left). The module's RXD port receives the UART data and automatically transmits it via radio waves over the air to the remote Bluetooth module. The remote Bluetooth module (right) receives the airborne data and delivers it via its TXD pin to the local device, like an MCU (right).

3.2 MCU-to-Module Communication



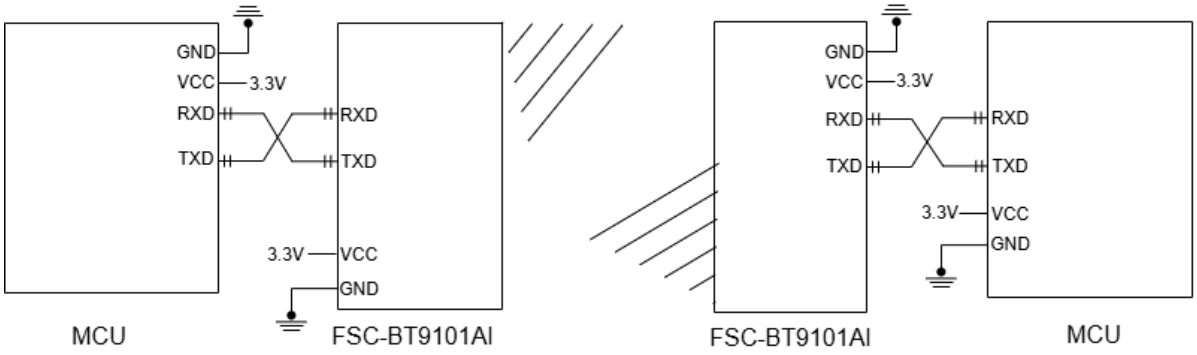
This diagram illustrates the connection between a master MCU (Microcontroller Unit) and an FSC-BT9101AI Bluetooth module via cross-connected serial ports, enabling command inter-

action between the master and the Bluetooth module to support wireless communication functions. It is applicable to IoT devices, remote control, and other scenarios.

1. **Serial Communication Interface**: The master MCU’s transmit pin (MCU_TX) is cross-connected to the Bluetooth module’s receive pin (UART_RX), and the master MCU’s receive pin (MCU_RX) is similarly connected to the Bluetooth module’s transmit pin (UART_TX), forming a bidirectional data transmission channel.
2. **Power and Grounding**: The Bluetooth module is powered by 3.3V through the VDD_3V3 pin and shares a common ground (GND) with the master MCU, ensuring level compatibility and signal stability.

3.3 Module-to-Module Communication

Two FSC-BT9101AI Bluetooth modules can establish a Bluetooth connection automatically upon power-up.



A module can act as a master device to connect to a slave device. The host can send commands to control the module for Bluetooth scanning, connection establishment, data transmission, and connection termination.

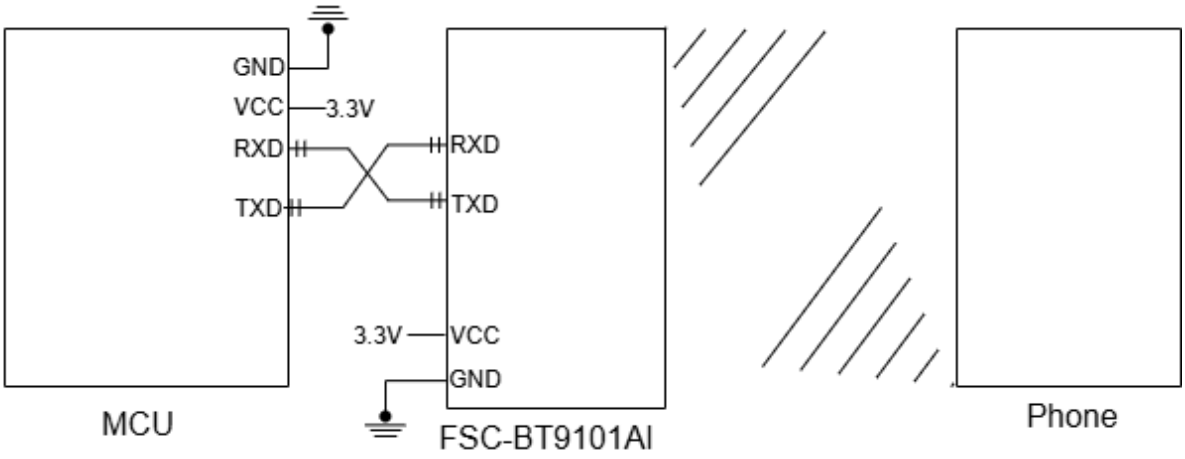
3.4 Module-to-Phone Communication

3.4.1 Why do we need to use an app on a mobile phone for Bluetooth connection and communication?

Native phone Bluetooth functionality primarily supports common use cases like audio transfer and file sharing. Some Bluetooth peripheral devices can be connected via the phone's built-in settings (e.g., Bluetooth speakers, headphones, keyboards, mice). However, when a peripheral device, like a module only supporting SPP/GATT protocols,

cannot be connected via native phone settings, a specific mobile application, such as the FeasyBlue app, is generally required for connection.

3.4.2 Communication Application



Bluetooth Module Side（FSC-BT9101AI）： Continuously transmits broadcast data after power-up.

Mobile Terminal : Can discover the broadcast packets via scanning and initiate a connection request to the module (FSC-BT9101AI). Upon successful connection, the Bluetooth module (FSC-BT9101AI) will pull the connection status pin HIGH and respond to status indication commands (valid in Command Mode) to notify the host of the successful Bluetooth connection.

Host: Can send data to the remote (Mobile Terminal) Bluetooth via the UART through the Bluetooth module. Conversely, the remote (Mobile Terminal) Bluetooth can also send data back to the host.

Chapter 4

Quick Development Kit

4.1 Datasheet

- FSC-BT9101 Datasheet
- FSC-BT9104 Datasheet

4.2 Evaluation Board

- FSC-DB005：Feasycom USB to Serial Bluetooth Data Transmission Application Development Board.

4.3 AT Command Set

- FSC-BT910x General Dual-Mode Data AT Command Set：For FSC-BT910x General Dual-Mode Data Transmission Application Firmware.
- FSC-BT910x Printer Multi-link AT Command Set：For FSC-BT910x Printer Multi-link Application Firmware.

4.4 Serial Port Tool

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

4.5 App&SDK

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

4.6 Firmware Upgrade

4.6.1 OTA Upgrade

- Tool : FeasyBlue App (Base on Android or iOS)
- Upgrade Guide : Please refer to FSC-BT910x - OTA Upgrade.

4.6.2 UART Upgrade

- Tool : Feasycom UART Upgrde Tool (Base on Windows PC)
- Upgrade Guide : Please refer to FSC-BT910x - UART Upgrade.

Chapter 5

Quick Start

[中文版]

Taking the FSC-BT9101AI module as an example:

5.1 What you need

5.1.1 Required Hardware

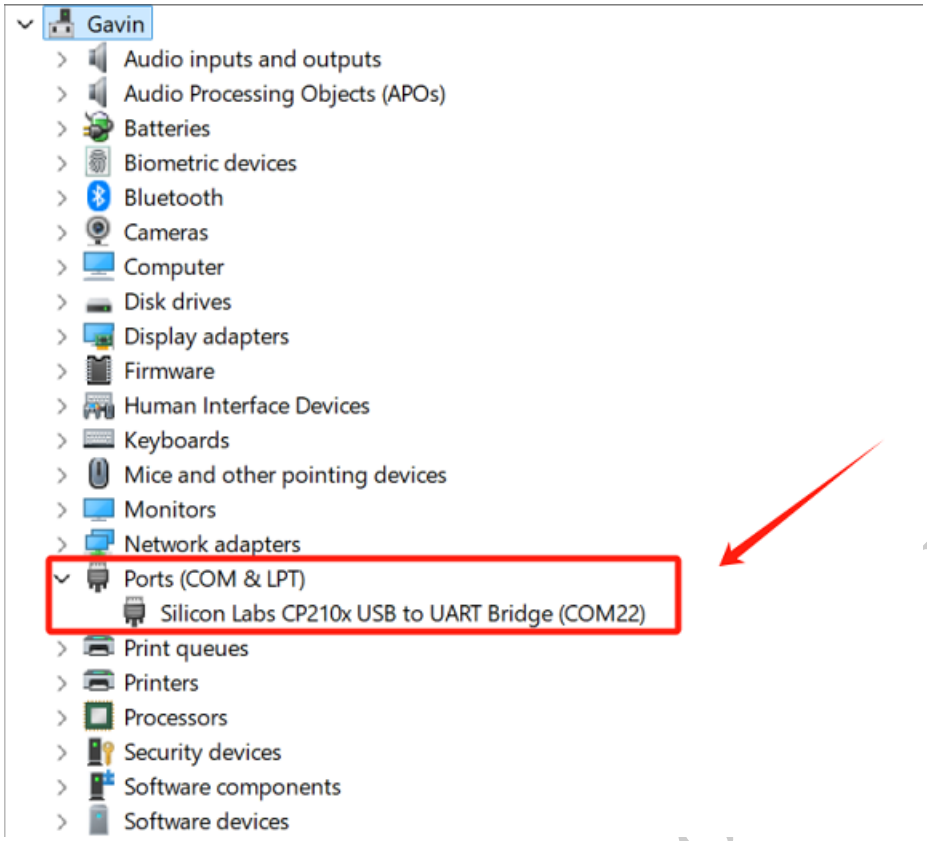
- 1 x FSC-DB005-BT9101AI : FSC-DB005 USB-to-Serial Rapid Evaluation Board with Integrated FSC-BT9101AI.
- 1 x PC（Windows / Mac）
- 1 x Phone（Android / iOS ）

5.1.2 Software and Setup

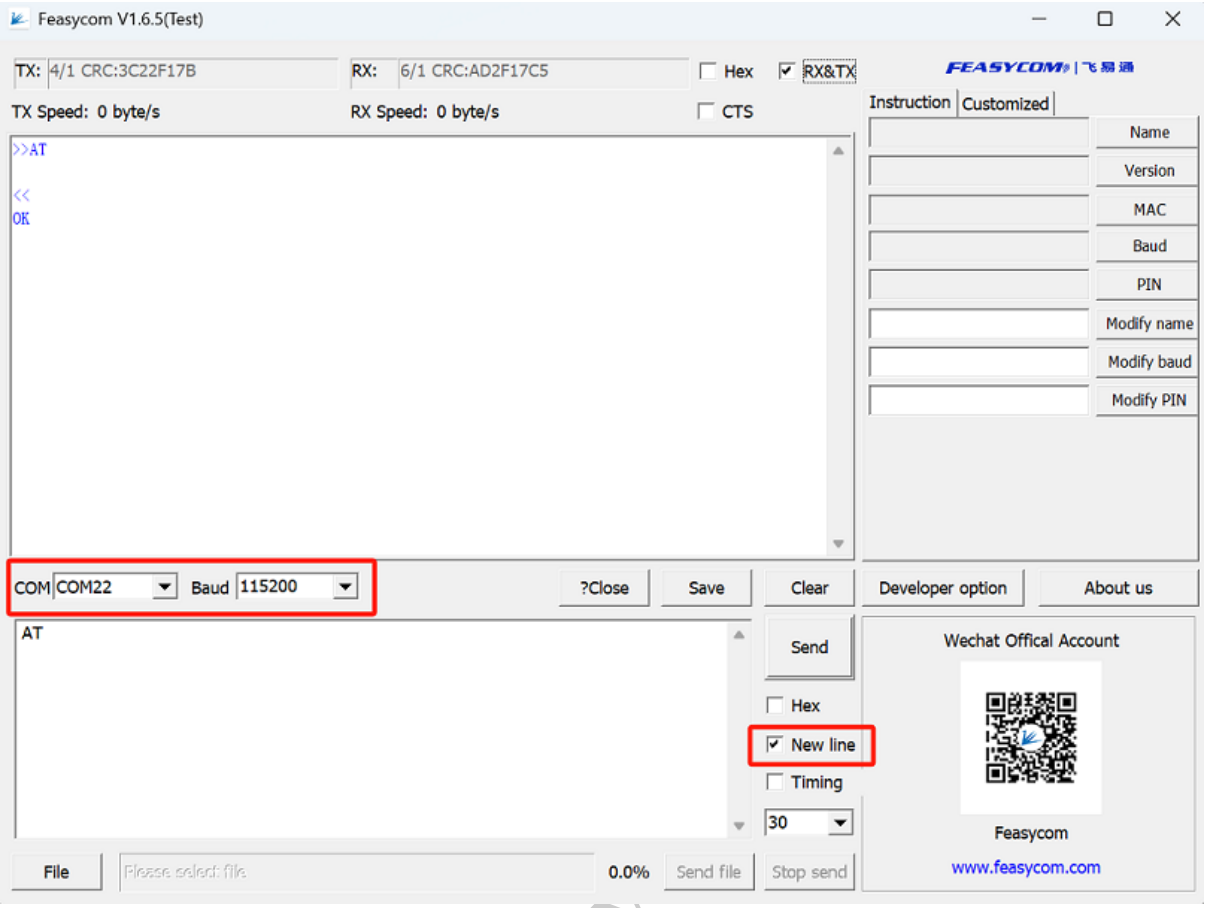
- Feasycom Serial Port Tool : A serial port communication analysis tool based on Windows PC.
- Communication Interface : UART
- UART Configuration：115200/8/N/1

5.2 Hardware Access

1.Connect the FSC-DB005-BT9101AI to the PC via USB. The PC will automatically recognize the serial port and generate a virtual COM port.



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



5.3 Communication Test

The following are several basic general AT command test examples.

For more AT commands, please refer to FSC-BT910x General Dual-Mode Data AT Command Set .

5.3.1 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.
```

5.3.2 AT+NAME - Read/Write Local BR/EDR Name

Example: Read BR/EDR Name

```
send:     <<AT+NAME\r\n
response: >>\r\n+NAME=FSC-BT9101-XXXX\r\n   //Example, please refer␣
↪to the actual reading result
response: >>\r\nOK\r\n
```

5.3.3 AT+LENAME - Read/Write Local BLE Name

Example: Read BLE Name

```
send:     <<AT+NAME\r\n
response: >>\r\n+NAME=FSC-BT9101-LE-XXXX\r\n   //Example, please␣
↪refer to the actual reading result
response: >>\r\nOK\r\n
```

5.3.4 AT+VER - Read Current Firmware Version

Example: Read Current Firmware Version

```
send:     <<AT+VER\r\n
response: >>\r\n+VER=1.0.0,FSC-BT9101\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-BT910x series dual-mode Bluetooth data transmission modules have two data transmission modes: **Throughput Mode** and **Command Mode**.

The generic data throughput firmware for the FSC-BT910x series modules default to throughput mode. To switch modes, refer to the FSC-BT910x General Dual-Mode Data AT Command Set and use the **AT+TPMODE** command.

The operation and differences between the two 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**:

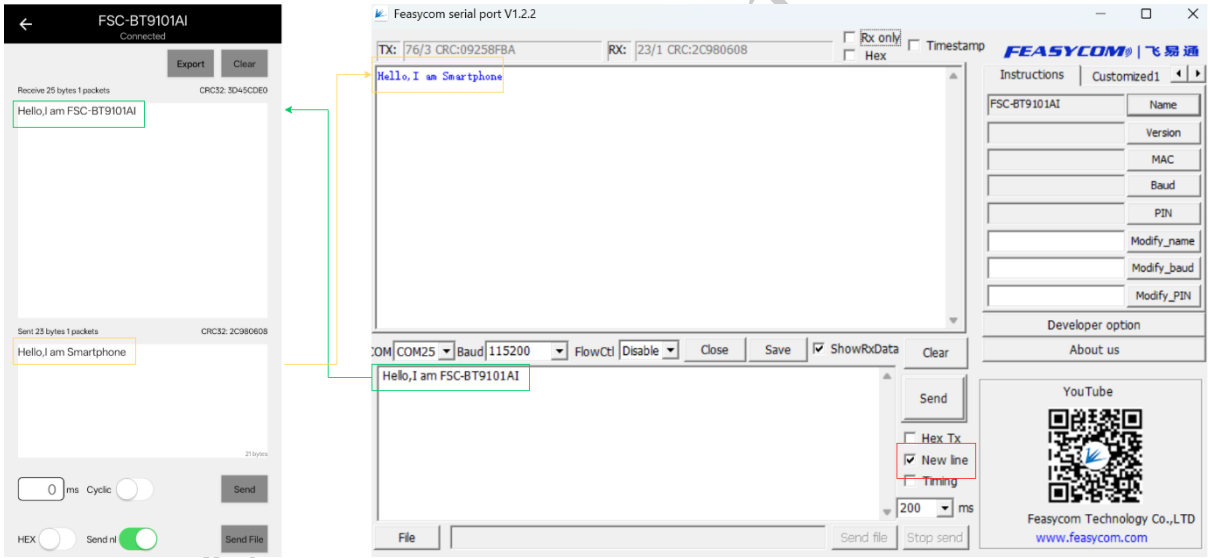
  Bluetooth Not Connected: Data received via UART is parsed as AT commands.

  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+SPPSEND** or **AT+LESEND** .

6.1.2 Module to Phone Application

1. **Module** : After power-on, the module will continuously send broadcast packet data.
2. **Phone** : Open the FeasyBlue APP , scan for nearby Bluetooth device advertisements, find the target Bluetooth module, and establish a connection.
3. **Connection Success** : After successful connection, the status pin of the module will pull up the level, indicating that the connection has been established.
4. **Data Transmission** : After successful connection, in the througput 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 SPP communication and data throughput transmission between an FSC-BT9101AI and an FSC-BT986 Bluetooth module, as follows:

1.Scan for nearby SPP devices

FSC-BT9101AI scans for nearby Bluetooth SPP devices, as follows:

```
Send:     <<AT+SCAN=1                    // Scan for nearby Bluetooth␣
↪SPP devices
```

(continues on next page)

(continued from previous page)

```
Response: >>OK
Response: >>+SCAN={                    // Scan started
Response: >>+SCAN=1,2,DC0D3100015E,-58,10,FSC-BT986
Response: >>+SCAN=2,2,DC0D30000016,-80,20,FSC-BT1038C-AKM-0016
Response: >>+SCAN=3,2,DC0D300022C1,-84,10,FSC-BT826F
Response: >>+SCAN=4,2,AC198EAC5DFE,-78,2,FR
Response: >>+SCAN=5,2,DC0D3053FFEE,-90,16,iMin TF2-34 FFEE
Response: >>+SCAN=6,2,1418C3B28AC1,-76,15,DESKTOP-U13VRNN
Response: >>+SCAN=7,2,DC0D3070F37A,-86,5,TT440
Response: >>+SCAN=8,2,A0B339DB5208,-66,10,QUZHENGWEI
Response: >>+SCAN=}                   // Scan ended
```

2.Establish SPP connection request

FSC-BT9101AI Establish an SPP protocol connection with the FSC-BT986 via the AT+SPPCONN command. The operation is as follows:

```
Send:     <<AT+SPPCONN=DC0D3100015E    // Initiate an SPP connection␣
↪request to the remote FSC-BT986
Response: >>OK
```

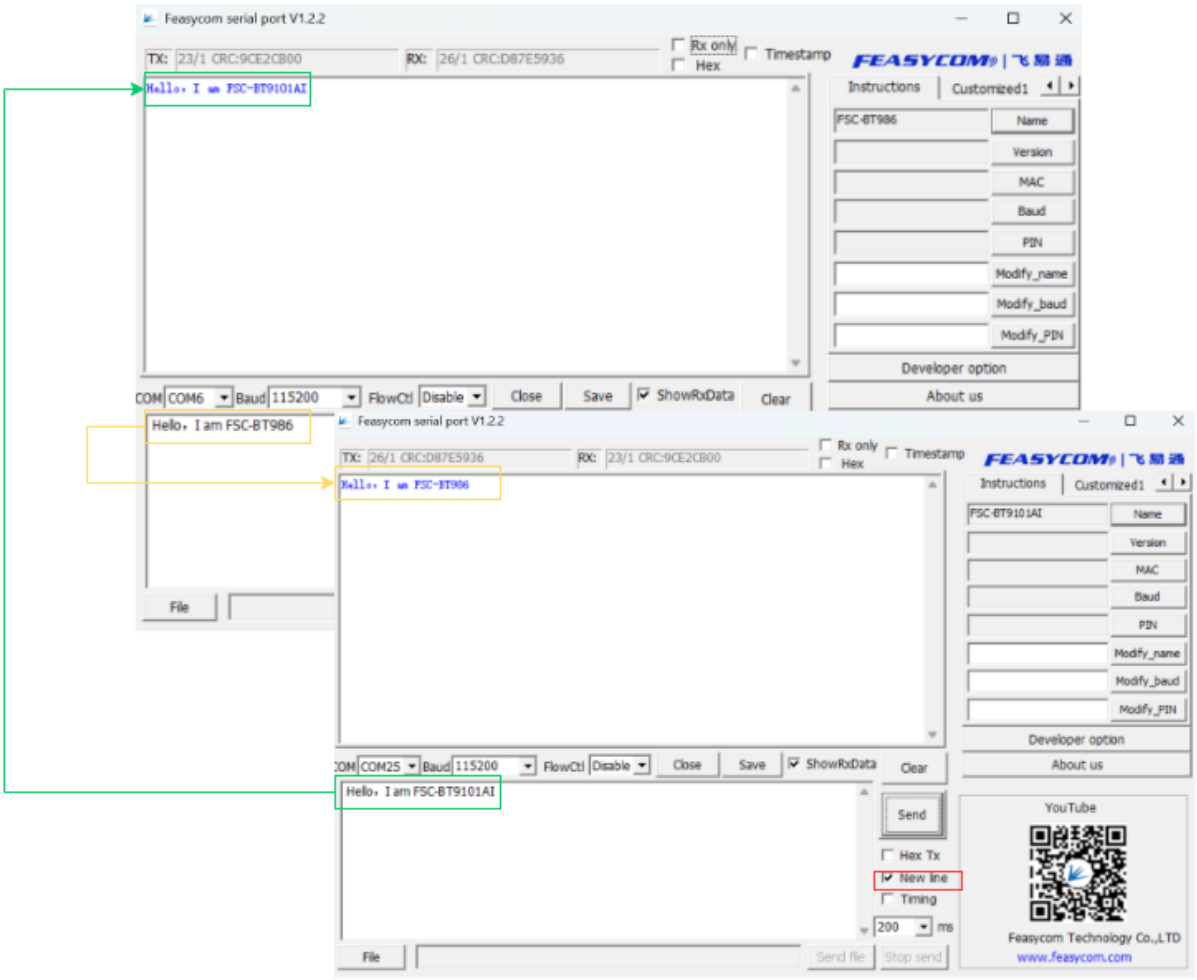
3.SPP Connected

In throughput mode, after a successful Bluetooth connection, the UART will not receive event response indicators. The connection status can be determined by the level state of the status pin (e.g., Pin 33) on the FSC-BT9101AI, as follows:

- **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

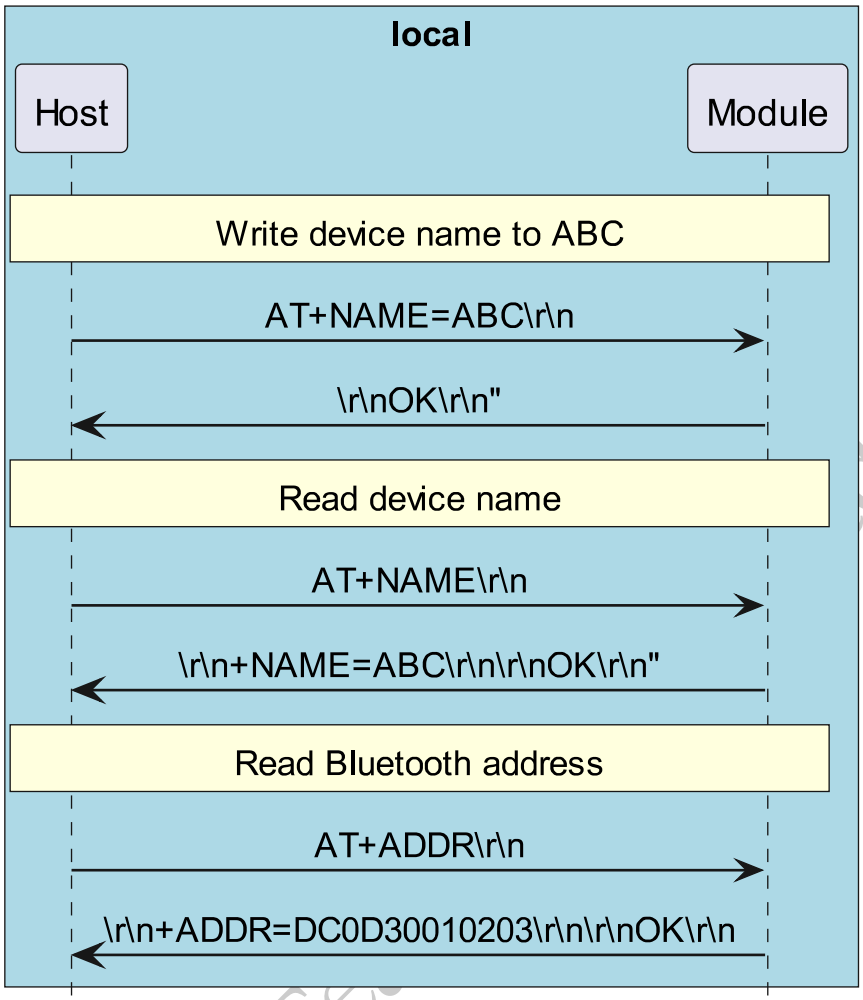
Throughput mode of the general data transmission firmware is enabled by default. After the SPP connection is successfully established, data transparent transmission communication can be performed, as shown in the following figure:



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. As follow:

1. Write the device name to ABC
2. Read the device name
3. Read the Bluetooth address

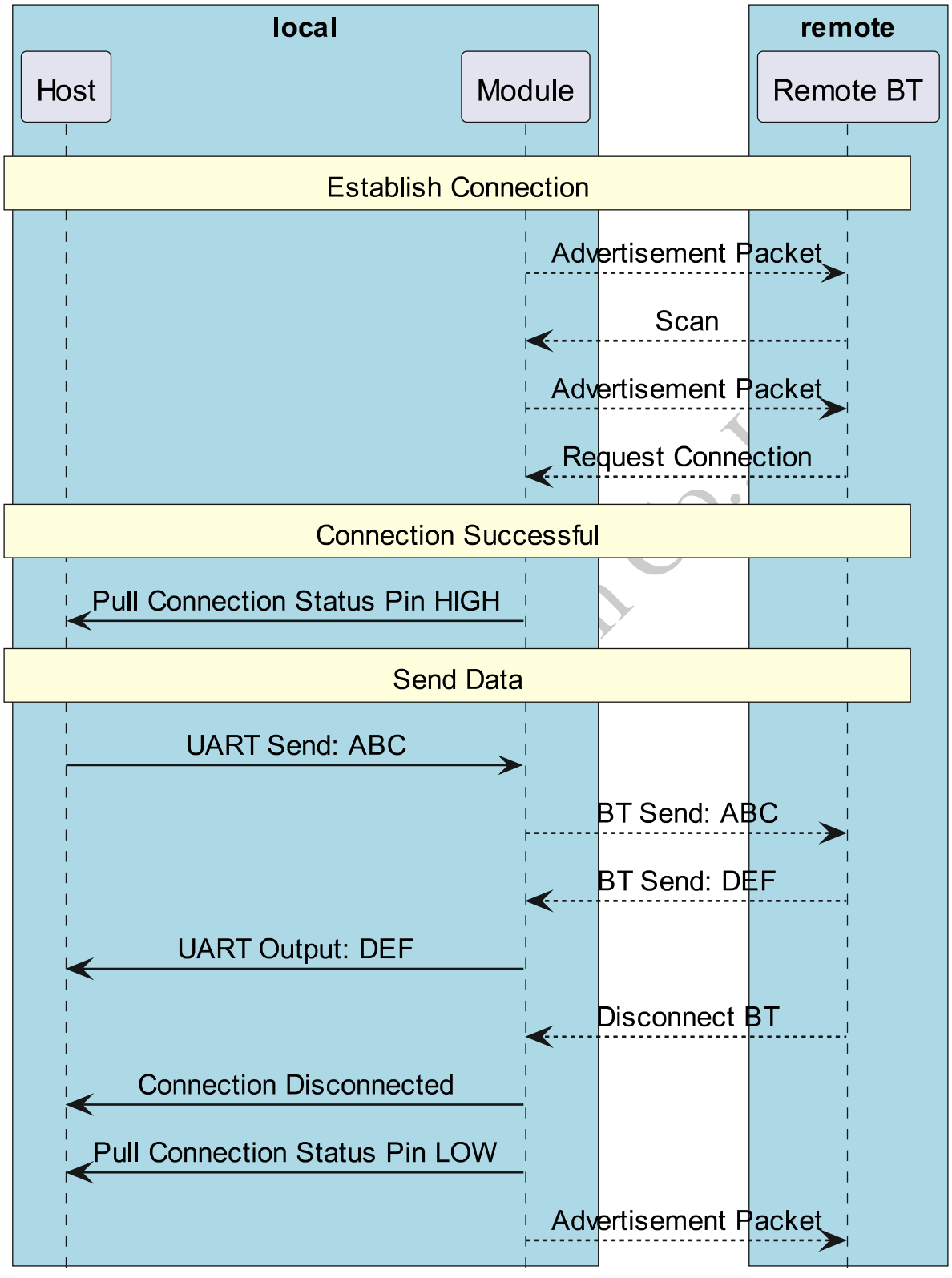


6.3 Data Transmission Flow

Upon power-up, the module continuously transmits advertisement data. A remote Bluetooth device (e.g., phone) can discover these advertisement packets via scanning and initiate a connection request to the module.

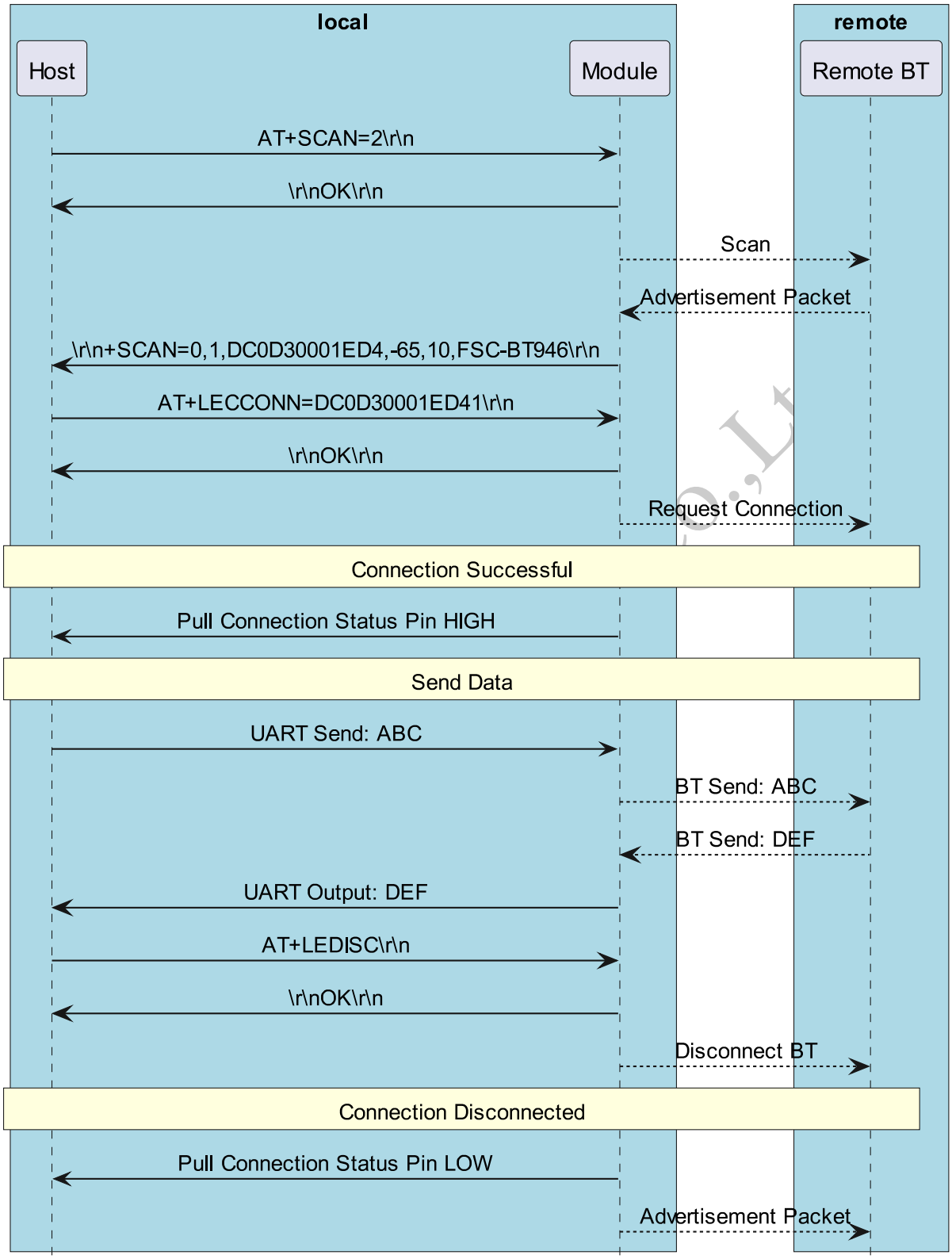
Upon successful connection, the module pulls its connection status pin HIGH to notify the host of the successful Bluetooth connection.

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 a master device to connect to a slave device. The host can send commands to control the module to perform scanning, connection, and disconnection.


local
remote
Host
Module
Remote BT
AT+SCAN=2\r\n
\r\nOK\r\n
Scan
Advertisement Packet
\r\n+SCAN=0,1,DC0D30001ED4,-65,10,FSC-BT946\r\n
AT+LECCONN=DC0D30001ED41\r\n
\r\nOK\r\n
Request Connection
Connection Successful
Pull Connection Status Pin HIGH
Send Data
UART Send: ABC
BT Send: ABC
BT Send: DEF
UART Output: DEF
AT+LEDISC\r\n
\r\nOK\r\n
Disconnect BT
Connection Disconnected
Pull Connection Status Pin LOW
Advertisement Packet

Chapter 7

Firmware Upgrade

[中文版]

7.1 OTA Upgrade

7.1.1 Tool

- FeasyBlue App （Based on Android & iOS）

7.1.2 User Guide

1.Run the FeasyBlue App, select **Settings - OTA Upgrade** to enter the OTA Upgrade function section. After entering, you will navigate to the device search interface to select the Bluetooth device to be upgraded.

2.Search for and select the Bluetooth device that needs upgrading. After selecting the device, you will enter the **Load Firmware** function interface to load the firmware upgrade file.

3.There are two ways to load the firmware upgrade file:

- **Select Firmware**: Choose to load the firmware upgrade file stored in the mobile phone's local storage (the firmware upgrade file is provided by Feasycom).
- **Get Firmware**: Enter the DFU name to download and import the corresponding firmware upgrade file from the cloud server via the network (the DFU name is provided by Feasycom).

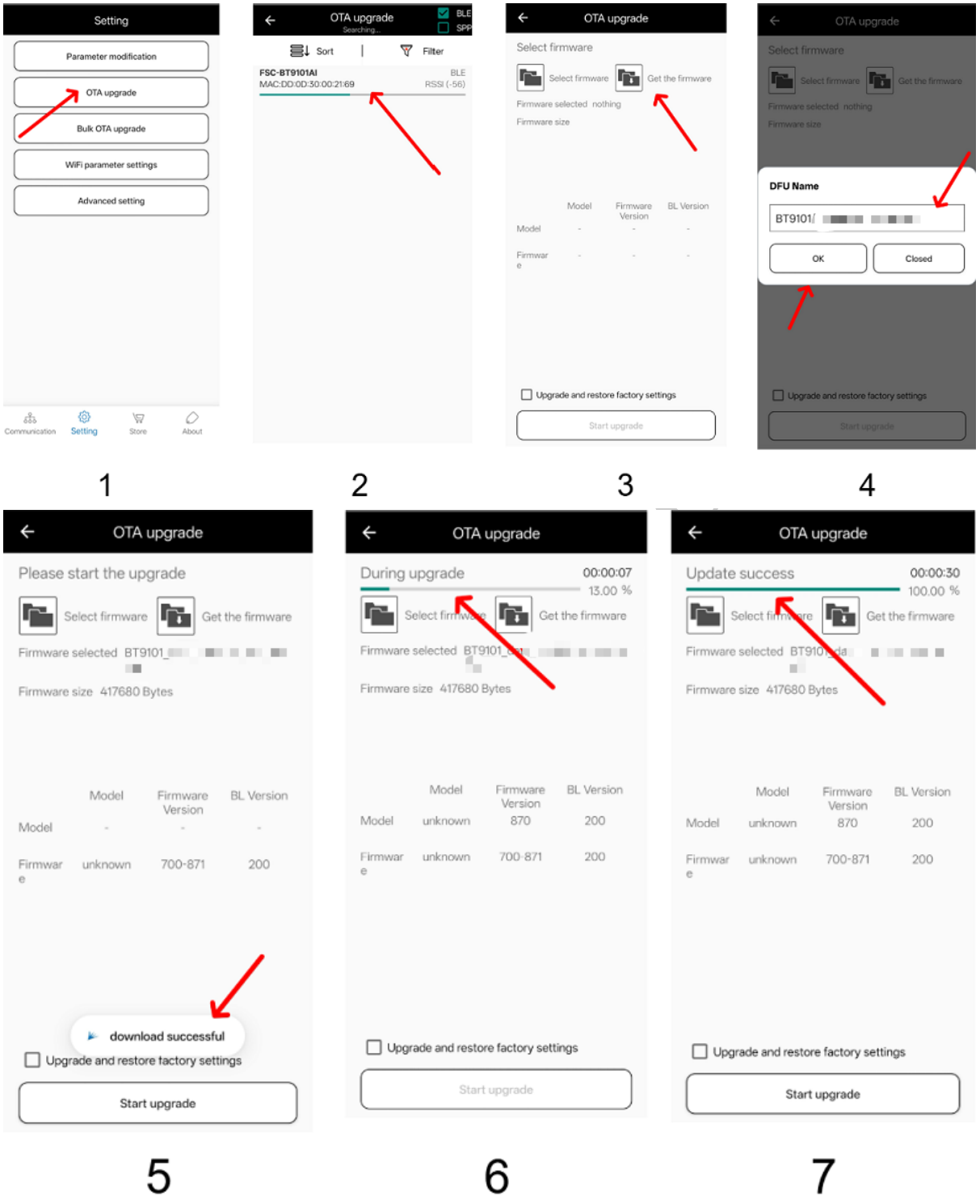
4.After the firmware file is loaded successfully, click the **Start Upgrade** button. The interface will display "Upgrading" and the upgrade progress, indicating that the upgrade mode is activated and the upgrade is in progress.

5.Wait for the upgrade progress bar to complete and the interface to display “Upgrade Completed”—the upgrade is then finished.

Warning :

1. When installing and running the FeasyBlue App, please allow the App to access the permissions for **nearby devices**, **location information**, and **media and file access**. Otherwise, you may fail to search for nearby Bluetooth devices and load the firmware upgrade file.
2. If you use the “Obtain Firmware” method (importing the firmware upgrade file by entering the DFU name), note that the mobile phone must be connected to the Internet, and ensure the DFU name is entered correctly (case-sensitive). Otherwise, an error of “network or file error” may occur.
3. Do not disconnect the power during the upgrade process.

7.1.3 Example



7.2 UART Upgrade

7.2.1 Tool

- Feasycom UART Upgrde Tool : Based on Windows PC

7.2.2 User Guide

1.Run the Feasycom UART Upgrde Tool.

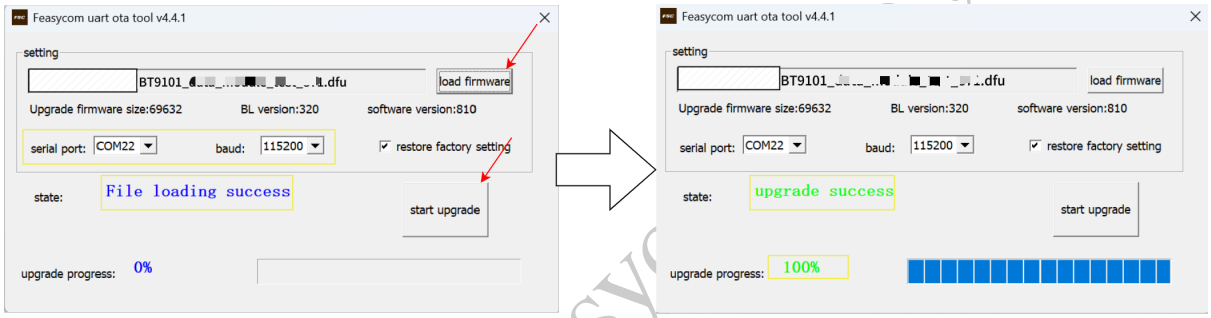
2.Enter the correct parameter configuration: **serial port**, **baud** .

3.Click **load firmware** and select the **.dfu firmware upgade file** stored locally on the PC.

4.Click **start upgrade** and will display **connection Successful** and simultaneously show the **upgrade progress**, indicating entry into upgrade mode.

5.When the progress bar displays **100%** and the status shows **upgrade success**, the serial port upgrade is complete.

7.2.3 Example



Chapter 8

FAQs

[中文版]

8.1 Why is a dedicated App needed for Bluetooth connection and communication on a phone?

Native phone Bluetooth functionality primarily supports common use cases like audio transfer and file sharing. Some Bluetooth peripheral devices can be connected via the phone's built-in settings (e.g., Bluetooth speakers, headphones, keyboards, mice). However, when a peripheral device, like a module only supporting SPP/GATT protocols, cannot be connected via native phone settings, a specific mobile application, such as the FeasyBlue app, is generally required for connection.

8.2 How to get the Bluetooth MAC address on an iOS phone?

Due to security considerations, the iOS system converts the Bluetooth MAC address into a UUID at the underlying level before presenting it to upper-layer applications. Therefore, apps cannot directly obtain the device’s actual MAC address.

The FSC-BT910x series Bluetooth modules include the MAC address in their broadcasts by default. Apps can retrieve the MAC address from the advertisement packet using the following method.

```
- (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 Subsequent parameters are the Bluetooth
address
                return true
            }
        }else if([key isEqualToString:@"kCBAdvDataLocalName"])
        {
            //there is name
            //NSString *szName = [dict objectForKey: key];
        }
    }
```

(continues on next page)

(continued from previous page)

```
    return false;
}
```

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

[Download PDF]