FSC-BT691 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 **4**

2.1 Default Configuration 4

2.2 GPIO Indications 4

2.2.1 LED 4

2.2.2 BT Connection Status 4

2.2.3 Working Mode 5

2.2.4 Disconnect 5

2.3 Working Mode 5

2.3.1 Throughput Mode 5

2.3.2 Command Mode 5

2.4 Low-Power Strategy 5

2.4.1 Low-Power Mode Setting 6

2.4.2 Low-Power Operation Strategy 6

2.5 GATT Service 6

2.6 Data Transmission Rate 6

3 Data Communication Principles **7**

3.1 Working Principle 7

3.2 MCU-to-Module Communication 8

3.3 Module-to-Module Communication 9

3.4 Module-to-Phone Communication 9

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

3.4.2 Communication Application 10

4 Quick Development Kit **12**
4.1 Datasheet 12
4.2 Evaluation Board 12
4.3 AT Command Set 12
4.4 Serial Port Tool 12
4.5 App&SDK 12
4.6 Firmware Upgrade 13
4.6.1 OTA Upgrade 13

5 Quick Start **14**
5.1 What you need 14
5.1.1 Required Hardware 14
5.2 Software and Setup 14
5.3 Hardware Access 14
5.4 Communication Test 16
5.4.1 AT - UART Communication Test 16
5.4.2 AT+NAME - Read/Write Bluetooth Name 16
5.4.3 AT+VER - Read Current Firmware Version 16

6 Development Examples **17**
6.1 Data Throughput Mode Application 17
6.1.1 What is Throughput Mode? 17
6.1.2 Module to Phone Application 18
6.1.3 Module to Module Application 18
6.2 Read/Write Module Default Parameters 20
6.3 Data Transmission Flow 21
6.4 Module Acts as Master to Connect to Remote Device 22

7 Firmware Upgrade **24**
7.1 OTA Upgrade 24
7.1.1 Tool 24
7.1.2 User Guide 24
7.1.3 OTA Upgrade Show 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

This guide is applicable to:

FSC-BT691 Bluetooth BLE data throughput transmission module, which supports the BLE (GATT Client/GATT Server) Bluetooth protocol.

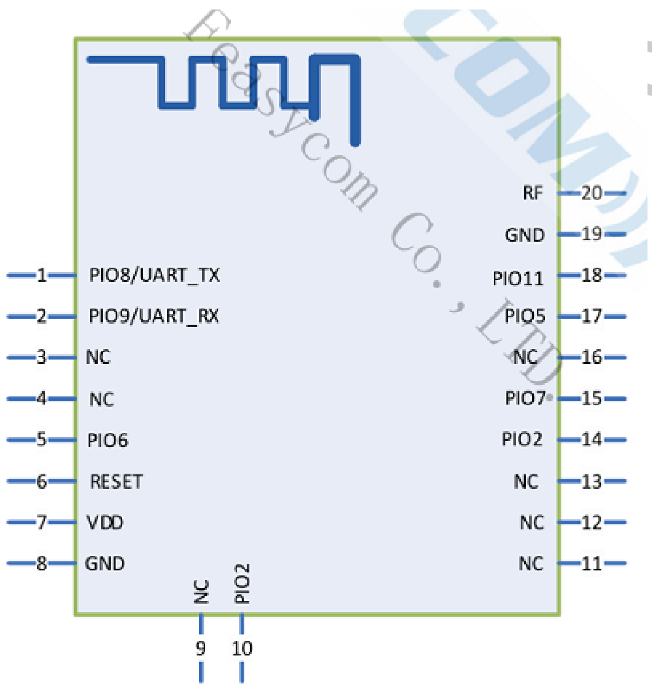
This guide consists of the following parts:

Chapter 1

Hardware Design

[中文版]

1.1 Module Pin Diagram



FSC-BT691 PIN Diagram(Top View)

1.2 Pin Description

Pin	Pin Name	Type	Pin Descriptions
1	UART_TX	O	UART transmit pin
2	UART_RX	I	UART receive pin
5	MODE	I	Low level for transparent transmission mode, high level for command mode
6	RESET	I	Low level for reset
7	VDD	Powe	3.3V power supply; LDO power supply is recommended
8	GND	GND	GND
15	DISCON-NECT	I/O	High level to disconnect (Bluetooth must be connected first)
17	LED	O	Outputs square wave when Bluetooth is disconnected; outputs high level when Bluetooth is connected
18	STATUS	O	Outputs low level when Bluetooth is disconnected; outputs high level when Bluetooth is connected
20	EXT_ANT	ANT	An external Bluetooth antenna can be connected by changing the 0-ohm resistor near the antenna

1.3 Hardware Design Note

- The module can be used by connecting only the VDD, GND, STATUS, UART_RX, and UART_TX pins.
- If the MCU needs to obtain the connection status of the Bluetooth module, the STATUS pin (Pin 18) must be connected.
- If the user wants to check the working status of Bluetooth, an LED light can be connected to the LED pin (Pin 17).
- After completing the schematic diagram, please send it to Feasycom for review to ensure the Bluetooth range reaches the optimal performance.

Chapter 2

Functional Description

[中文版]

2.1 Default Configuration

Name	Feasycom
UART Baudrate	115200/8/N/1

2.2 GPIO Indications

2.2.1 LED

Pin	Status	Description
Pin 17	Low Level	Not started
Pin 17	1 Hz square wave	Bluetooth disconnected
Pin 17	High Level	Bluetooth connected

2.2.2 BT Connection Status

Pin	Status	Description
Pin 18	Low Level	Bluetooth disconnected
Pin 18	High Level	Bluetooth connected

2.2.3 Working Mode

Pin 6	Status	Description
Pin 6	Low Level	Throughput Mode
Pin 6	High Level	Command mode

Note:

To use this function, the pin function must be enabled using the AT+PIOCFG command.

2.2.4 Disconnect

Pin	Status	Description
Pin 15	High Level	Disconnect Bluetooth connection

Note:

To use this function, the pin function must be enabled using the AT+PIOCFG command.

2.3 Working Mode

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+LESEND`.

2.4 Low-Power Strategy

2.4.1 Low-Power Mode Setting

Enable or disable the low power function through the command **AT+LPM{=Param}** .

2.4.2 Low-Power Operation Strategy

- **AT+LPM=1**: BT UART automatically enters low power mode if no data is received for more than 10 seconds. The first frame of data from the BT UART wakes up the Bluetooth. Once connected, it does not enter low power mode.

2.5 GATT Service

Type	UUID	Permissions	Description
Service	0xFFF0		Throughput service
Write	0xFFF2	Write,Write Without Response	APP to module
Notify	0xFFF1	Notify	Module to APP

2.6 Data Transmission Rate

Baudrate	Data Packet	Transmission Interval	Connection Interval	Transmission Mode	Rate
921600	182	20ms	7.5ms	Notify	4000 Byte/s

Chapter 3

Data Communication Principles

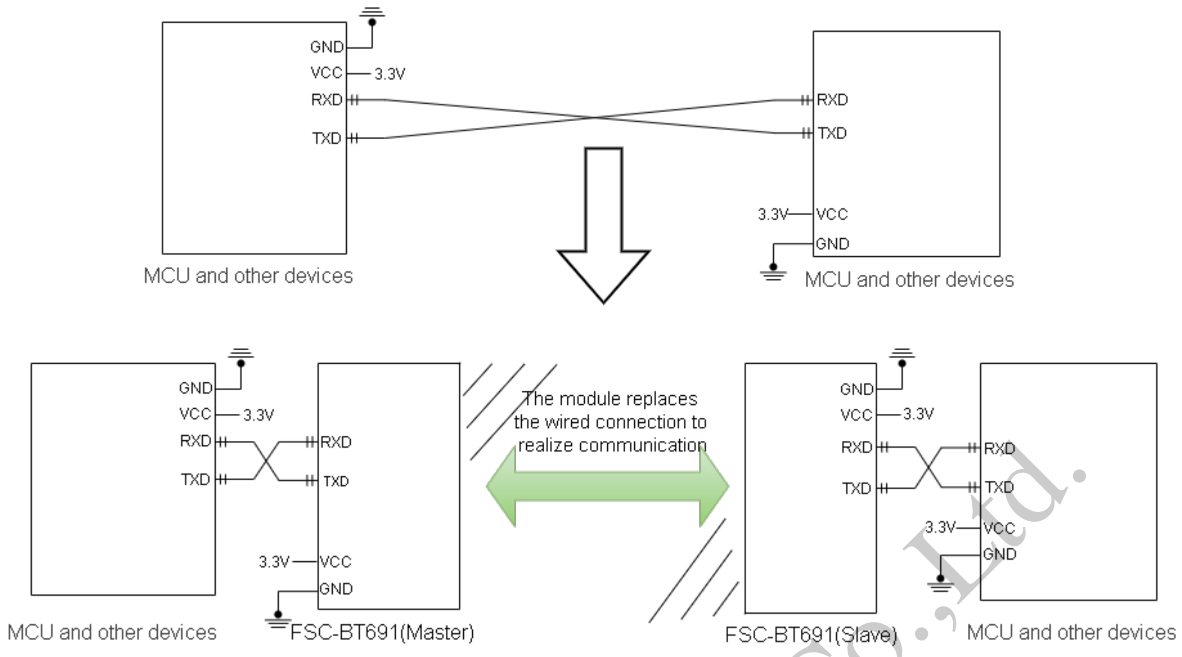
[中文版]

3.1 Working Principle

FSC-BT691 Bluetooth data transmission module realizes wireless communication between devices based on the BLE (Bluetooth Low Energy) protocol.

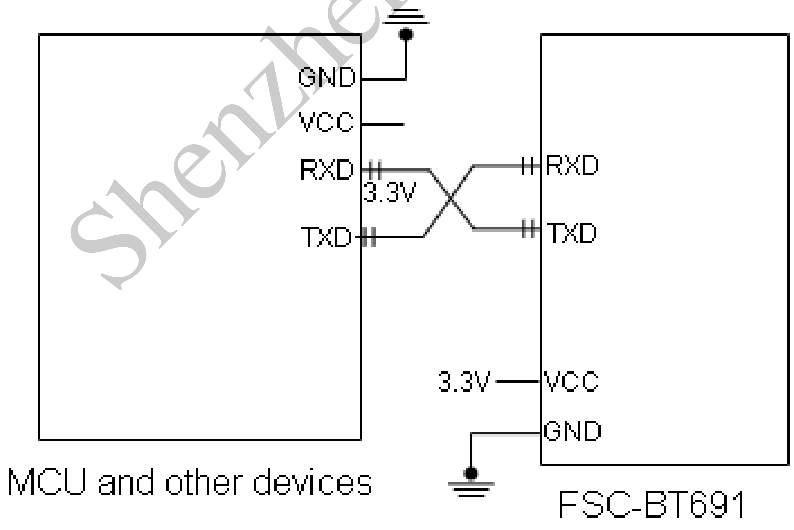
- **BLE**: Adopts an event-driven low-power architecture, and defines a “Service-Characteristic” model through the GATT protocol to realize intermittent small data interaction (such as sensor data), which is suitable for IoT devices.

The module can send AT commands or perform transparent data transmission with host devices (mobile phones/MCUs) via UART to complete connection establishment, data exchange, and status management.



As shown in the figure above, the FSC-BT691 Bluetooth module is used to replace physical wires in full-duplex communication. Devices such as MCU (left) send data to the FSC-BT691 Bluetooth module (left) through TXD. After the RXD port of the Bluetooth module receives the UART data, it automatically transmits the data via radio waves over the air to the remote FSC-BT691 Bluetooth module. The remote Bluetooth module (right) receives the over-the-air data and sends it to local devices such as MCU (right) through TXD.

3.2 MCU-to-Module Communication

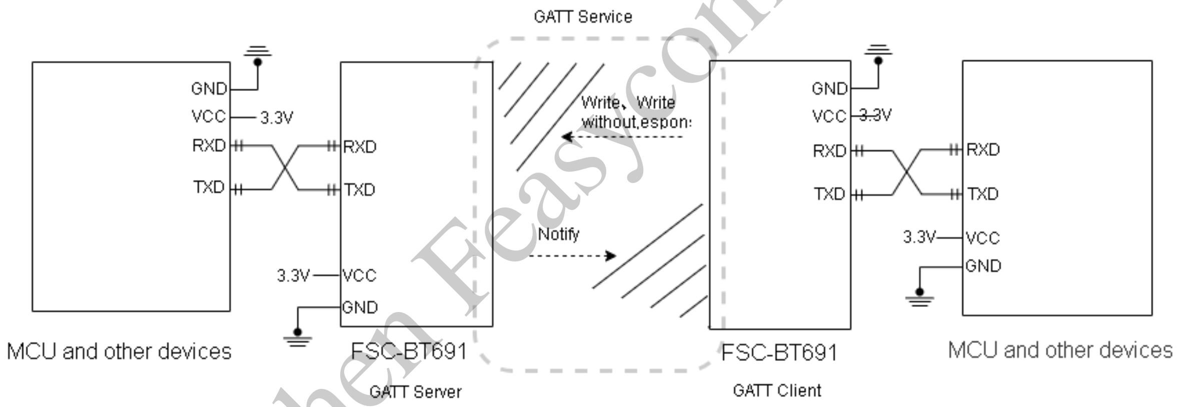


This diagram shows a connection schematic between a main control MCU (Microcontroller Unit) and an FSC-BT691 Bluetooth module. Command interaction between the main control and the Bluetooth module is realized through cross-connected UART, supporting wireless communication functions, which is applicable to scenarios such as IoT devices and remote control.

1. **UARTCommunication Interface：** The transmission end of the main MCU (MCU_TX) is cross-connected to the receiving end of the Bluetooth module (UART_RX), and the receiving end of the main MCU (MCU_RX) is similarly connected to the transmission end of the Bluetooth module (UART_TX), forming a bidirectional data transmission channel;
2. **Power Supply and Grounding** : The Bluetooth module is powered with 3.3V through the VDD_3V3 pin and shares a common ground (GND) with the main MCU to ensure level compatibility and signal stability;

3.3 Module-to-Module Communication

Two FSC-BT691 Bluetooth modules can establish a Bluetooth connection automatically after being powered on.



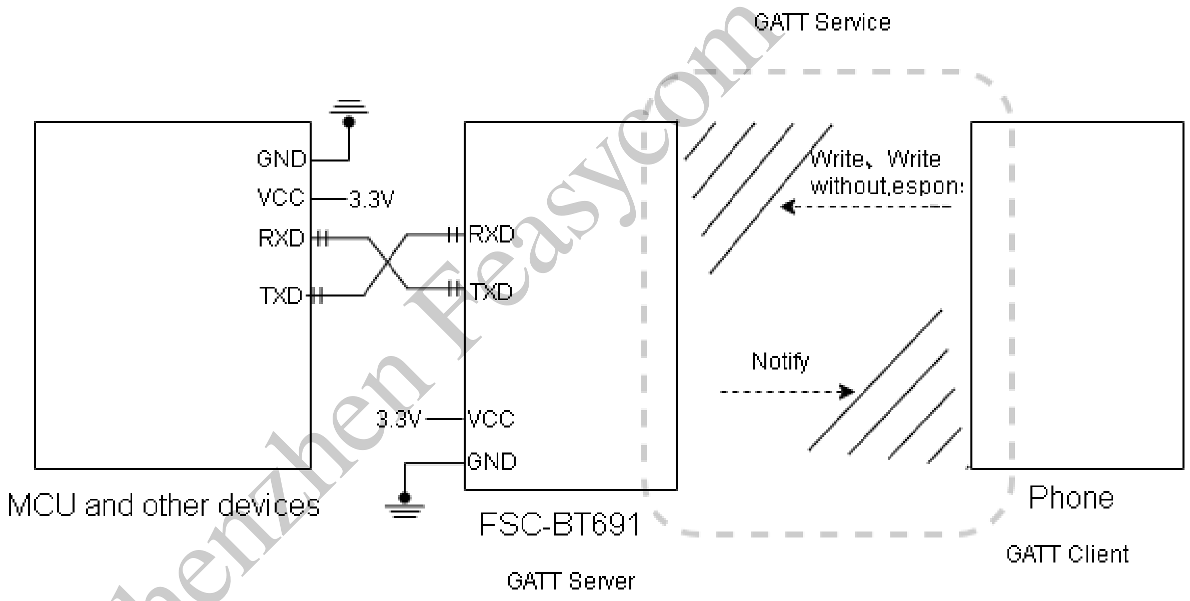
FSC-BT691 module has master-slave device functionality. The left module can be configured as a master device, and the right module as a slave device. The master device can implement operations such as Bluetooth scanning, connection establishment, data transmission, and disconnection by sending commands. Among them, the device that actively initiates a Bluetooth connection is defined as the master device, and the device that receives the connection request is the slave device.

3.4 Module-to-Phone Communication

3.4.1 Why is an APP required on the 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 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), a specific mobile application (such as the FeasyBlue app) is generally required to connect to such modules.

3.4.2 Communication Application



Bluetooth Module side (FSC-BT691) : It will continuously send broadcast data outward after being powered on.

Mobile Phone side : The FeasyBlue App can search and obtain the broadcast packet of the FSC-BT691 module, and initiate a MAC address/UUID connection request to the module (FSC-BT691), while acquiring all services and characteristics provided by the device. After a successful connection, the Bluetooth module (FSC-BT691) will pull high the connection status pin and report the connection status command (valid in command mode) to notify the host that the Bluetooth connection is successful.

Host side: Data can be sent to the remote (mobile phone side) Bluetooth through the UART via 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-BT691 Datasheet

4.2 Evaluation Board

- FSC-DB006： Feasycom Bluetooth Data Throughput Transmission Development Board.

4.3 AT Command Set

- FSC-BT691 General Data AT Command Set

4.4 Serial Port Tool

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

4.5 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.

4.6 Firmware Upgrade

4.6.1 OTA Upgrade

- Tool : SUOTA （Based Android）
- User Guide : Please refer to FSC-BT691 - OTA Upgrade

Chapter 5

Quick Start

[中文版]

5.1 What you need

5.1.1 Required Hardware

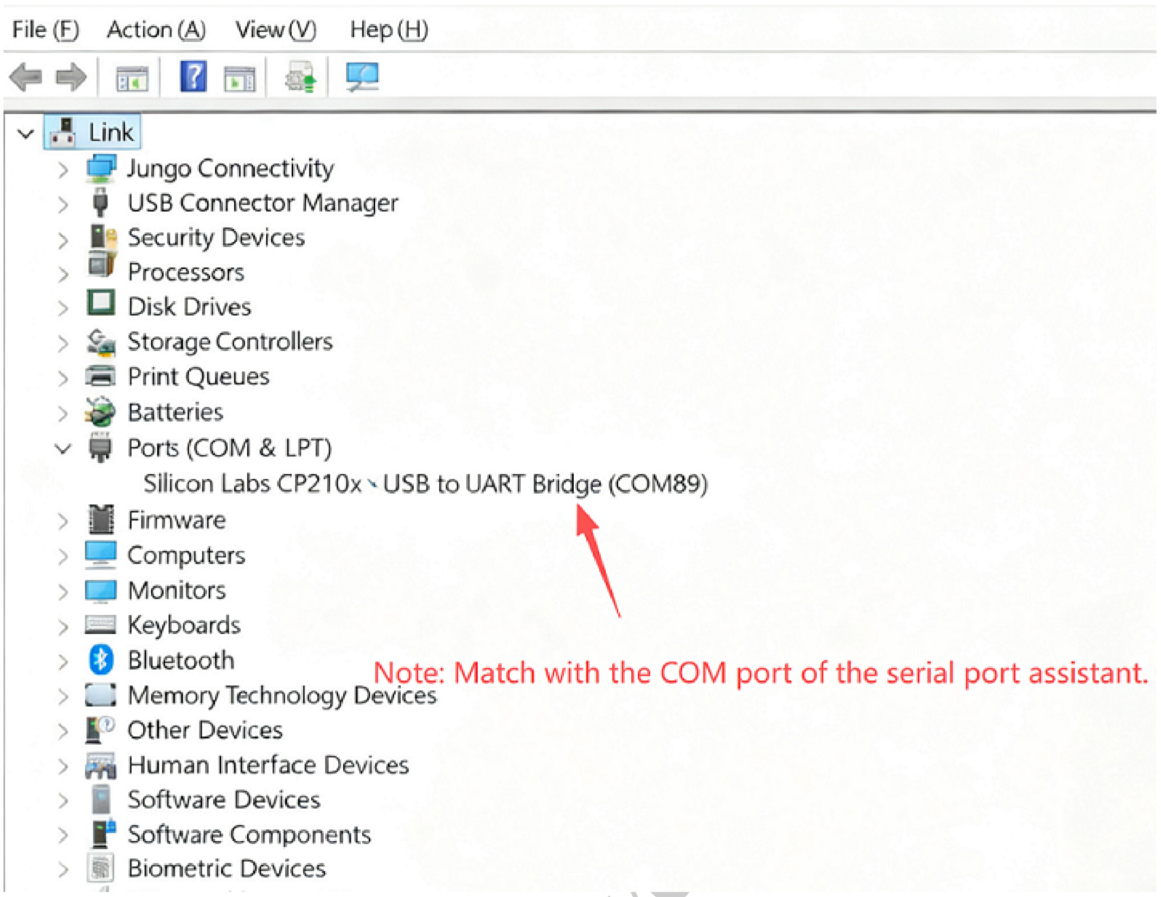
- 1 x FSC-DB006-BT691 Rapid Development Kit：FSC-DB006 USB-to-Serial Rapid Evaluation Board pre-integrated with FSC-BT691 module
- 1 x PC (Windows / Mac)

5.2 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

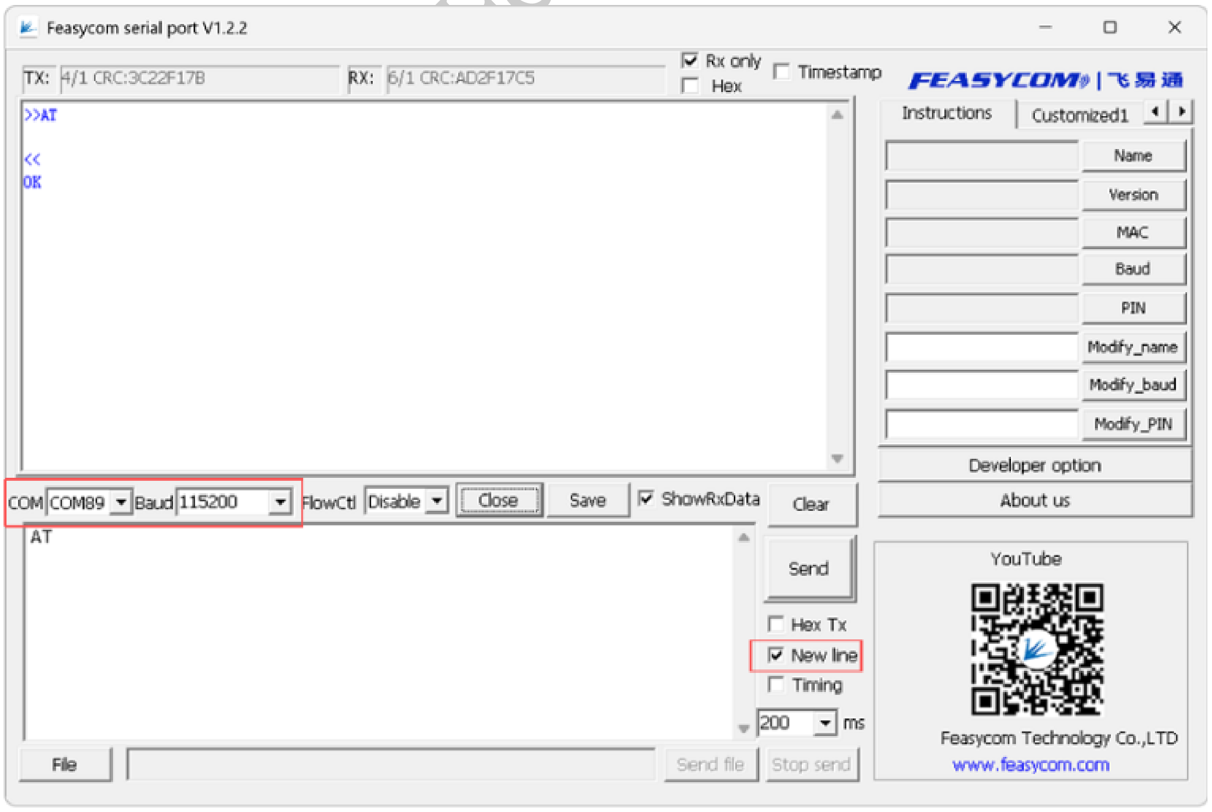
5.3 Hardware Access

1.Connect the FSC-DB006-BT691 Quick Development Kit to the PC via USB. The PC will automatically recognize the serial port and generate a virtual COMx port.



Note: Match with the COM port of the serial port assistant.

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



```
>>AT

<<
OK
```

COM COM89 Baud 115200

AT

200 ms

5.4 Communication Test

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

For more commands, please refer to FSC-BT691 General Data AT Command Set .

5.4.1 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.
```

5.4.2 AT+NAME - Read/Write Bluetooth Name

Example: Read Bluetooth name

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

5.4.3 AT+VER - Read Current Firmware Version

Example:

```
Send:     <<AT+VER\r\n
Response: >>\r\n+VER=1.0.0,FSC-BT691\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-BT691 Bluetooth BLE data module has two work modes: **Throughput Mode** and **Command Mode** .

Note

The master firmware only supports the throughput mode, while the slave firmware supports two modes: throughput Mode and command mode.

The generic data throughput firmware for the FSC-BT691 modules defaults to throughput mode. To switch modes, refer to FSC-BT691 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* .

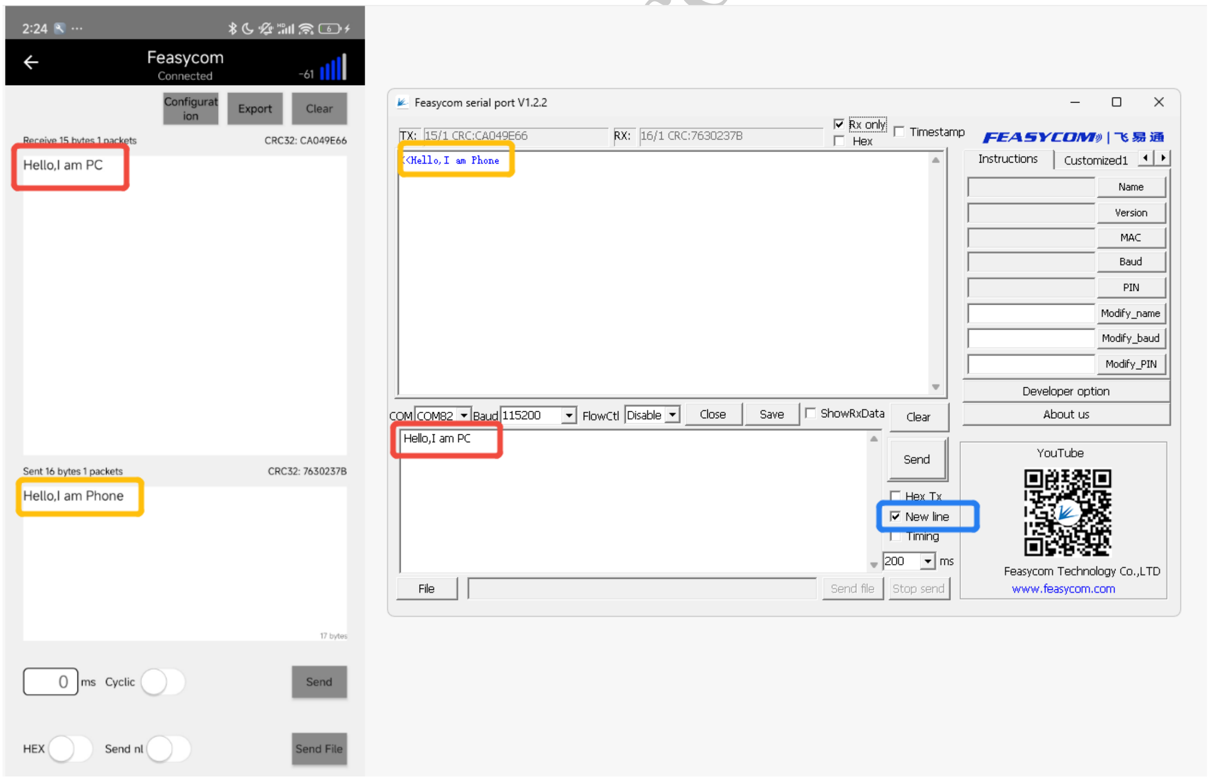
6.1.2 Module to Phone Application

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.


Receive 15 bytes 1 packets CRC32: CA049E66
Hello,I am PC
Sent 16 bytes 1 packets CRC32: 7630237B
Hello,I am Phone

Feasycom serial port V1.2.2
TX: 15/1 CRC:CA049E66 RX: 16/1 CRC:7630237B
<Hello,I am Phone
COM COM82 Baud 115200 FlowCtl Disable
Hello,I am PC

6.1.3 Module to Module Application

Demonstration of BLE communication data transparent transmission between FSC-BT691(Master) and FSC-BT3431(Slave) Bluetooth modules, as follows:

1.Scan for nearby BLE devices

FSC-BT691(Master) scans for nearby Bluetooth BLE devices:

```
Send: <<AT+SCAN=1                    // Scan nearby Bluetooth BLE devices
Response: >>OK
     >>+SCAN={                        //Scan Started
     >>+SCAN=0,DC0D30000A09,-84,8,Feasycom      // FSC-BT691(Slave)
     >>+SCAN=1,DD0D3040201C,-86,13,FSC-BW256B-LE
     >>+SCAN=0,E0798DB74D3C,-93,9,FSC-WY001
     >>+SCAN=1,DC0DAA7E5CBF,-86,10,FSC-BP106G
     >>+SCAN=1,559F3B6E9D2F,-85,15,LAPTOP-81TBTP4L
     >>+SCAN=}                       //Scan Completed
```

2.Establish BLE connection request

FSC-BT691(Master) establishe BLE protocol connection with FSC-BT691(slave) via the *AT+LECCONN* command as follows:

```
Send:     <<AT+LECCONN=DC0D30000A090     // 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 first parameter in the return result is the address type, as in the following example:

```
 //The address type is the first parameter, which is "0".

 Response: >>+SCAN=0,DC0D30000A09,-84,8,Feasycom
```

3.BLE Connection Established Successfully

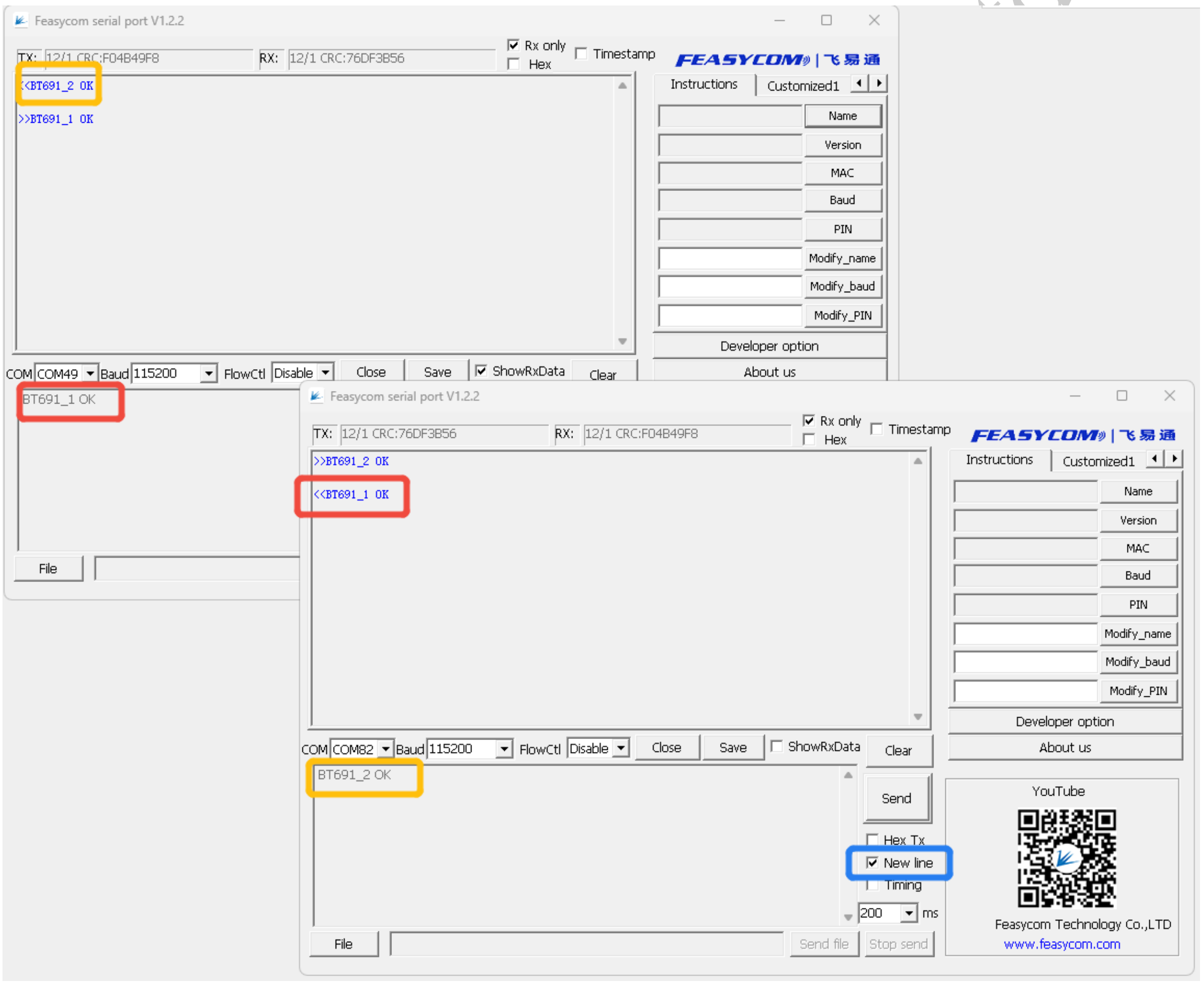
In throughput mode, after the Bluetooth connection is successfully established, the serial port cannot receive event response indicators. The current connection status can be determined by the level state of **Pin18** of FSC-BT691, as detailed below:

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:



Feasycom serial port V1.2.2

TX: 12/1 CRC:F04B49F8 RX: 12/1 CRC:76DF3B56 ☑ Rx only ☐ Hex ☐ Timestamp

```
<<BT691_2 OK
>>BT691_1 OK
```

COM COM49 Baud 115200 FlowCtl Disable ☑ ShowRxData

```
BT691_1 OK
```

Feasycom serial port V1.2.2

TX: 12/1 CRC:76DF3B56 RX: 12/1 CRC:F04B49F8 ☑ Rx only ☐ Hex ☐ Timestamp

```
>>BT691_2 OK
<<BT691_1 OK
```

COM COM82 Baud 115200 FlowCtl Disable ☐ ShowRxData

```
BT691_2 OK
```

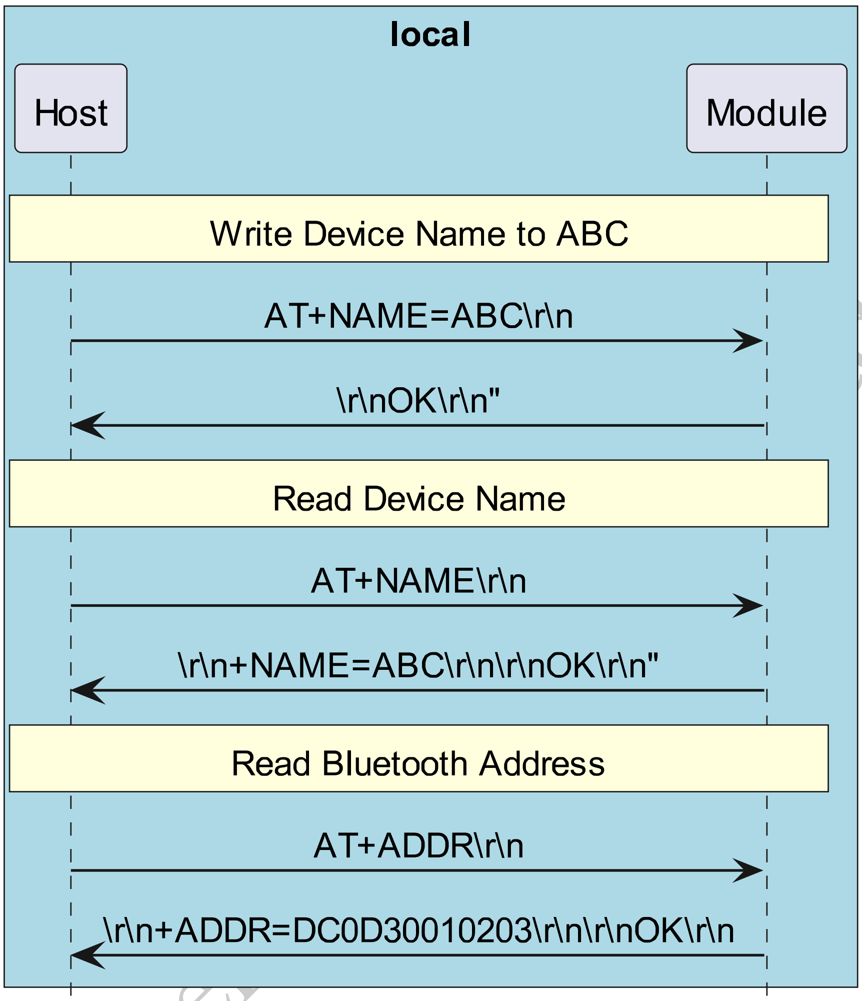
☐ Hex Tx ☑ New line ☐ Timing 200 ms

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

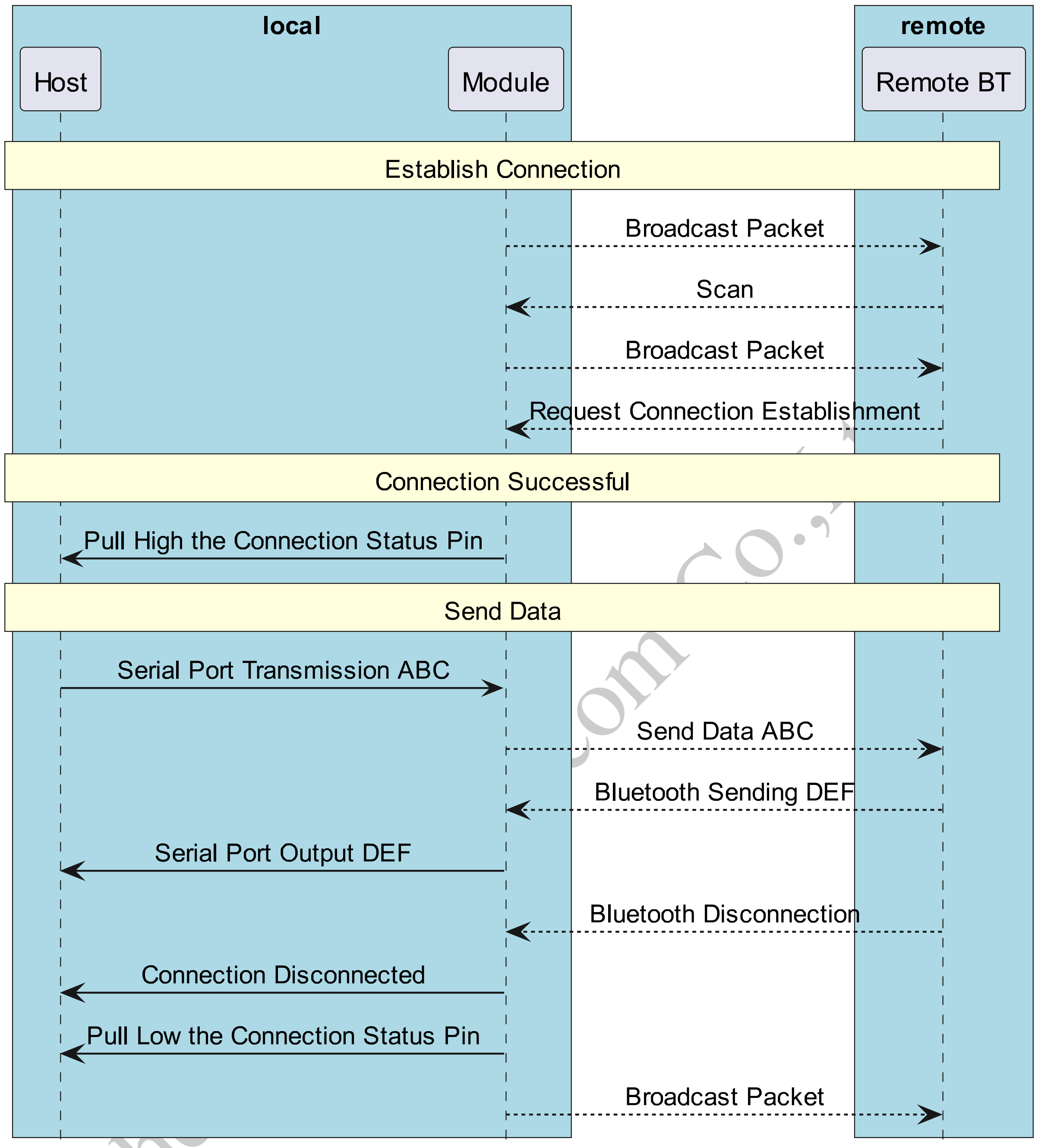


6.3 Data Transmission Flow

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.

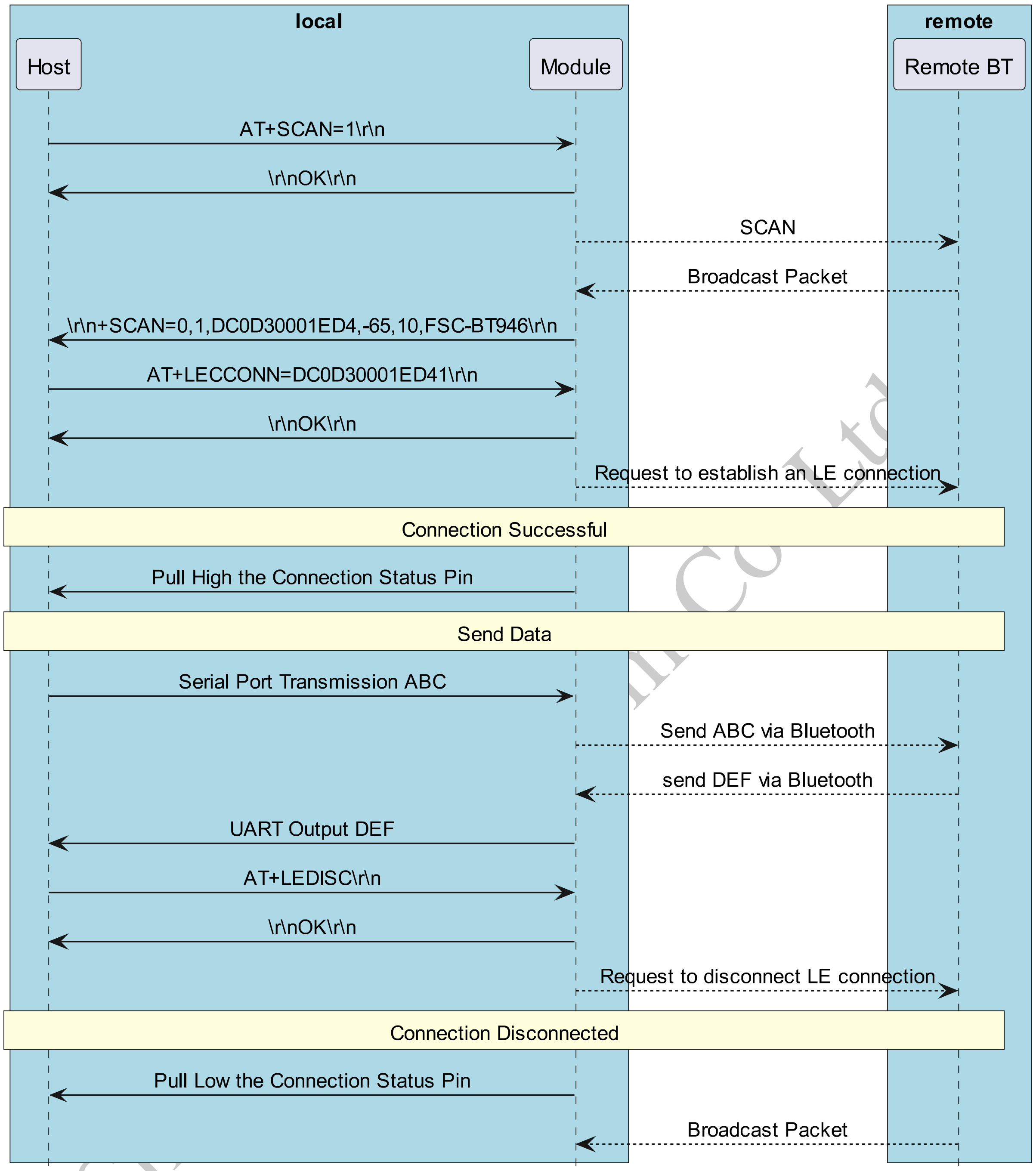
3.The host can send data to the remote Bluetooth device via the Bluetooth module, and the remote Bluetooth device can also send data to the host.



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

7.1.1 Tool

- SUOTA App

7.1.2 User Guide

1.Download and install the SUOTA App. After successful installation, the local mobile phone will automatically create a **Suota Folder**.

2.Store the firmware update file (.img file, provided by Feasycom) that needs to be updated in the Suota Folder automatically created by the SUOTA App. The App will automatically recognize the firmware update files in this folder.

3.Open the **SUOTA App**, tap the **SCAN** button to scan for and connect to the target Bluetooth device (e.g., Feasycom).

4.On the Device info page, tap **UPDATA DEVICE** to enter the **File selection** page.

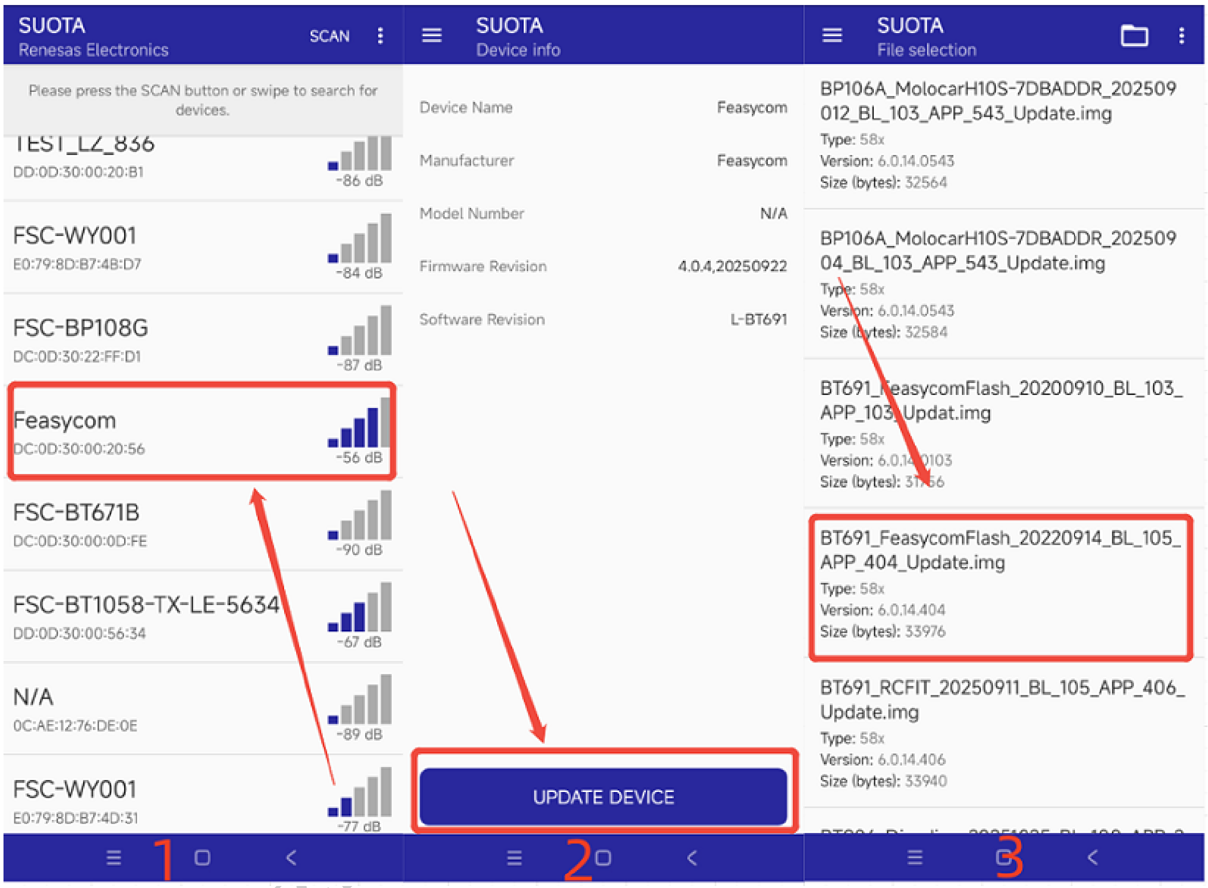
5.On the **File selection** page, select and import the firmware update file to be upgraded from the firmware upgrade files detected by the App. After successful import, it will navigate to the **Parameter settings** page.

6.On the **Parameter settings** page, modify the corresponding configurations according to the configuration parameters in **Operation Diagram 4 below**. After completion, tap the **SEND TO DEVICE** button to send the settings to the module and enter the update mode.

7.If the **Uploading file** page displays the update progress, it indicates that the update mode has been entered and the update has started.

8.When **Uploading file** page displayed **Upload Completed** , it indicates that the update is finished. Tap “OK” and reboot the device.

7.1.3 OTA Upgrade Show



Please press the SCAN button or swipe to search for devices.

TEST_LZ_836 DD:0D:30:00:20:B1 -86 dB
FSC-WY001 E0:79:8D:B7:4B:D7 -84 dB
FSC-BP108G DC:0D:30:22:FF:D1 -87 dB
Feasycom DC:0D:30:00:20:56 -56 dB
FSC-BT671B DC:0D:30:00:0D:FE -90 dB
FSC-BT1058-TX-LE-5634 DD:0D:30:00:56:34 -67 dB
N/A 0C:AE:12:76:DE:0E -89 dB
FSC-WY001 E0:79:8D:B7:4D:31 -77 dB

Device Name	Feasycom
Manufacturer	Feasycom
Model Number	N/A
Firmware Revision	4.0.4,20250922
Software Revision	L-BT691

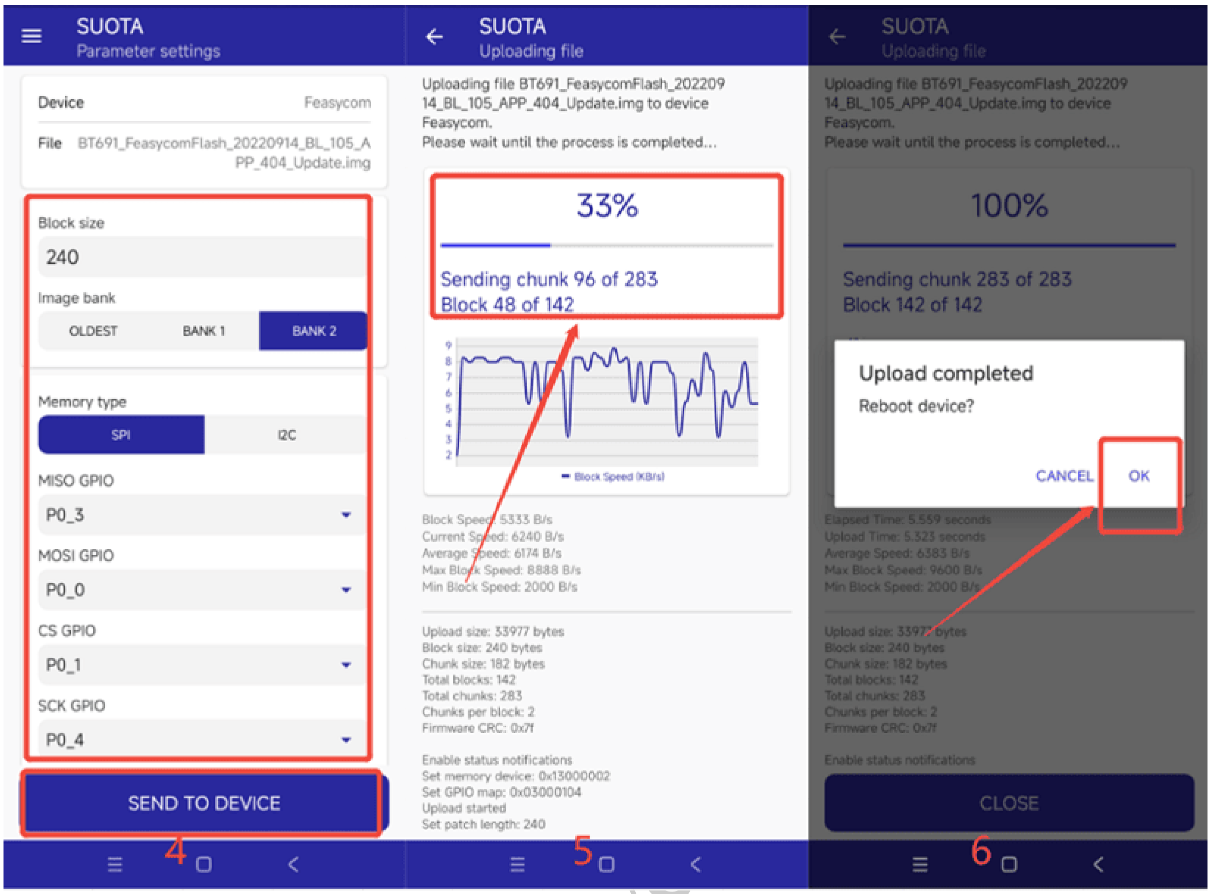
BP106A_MolocarH10S-7DBADDR_202509012_BL_103_APP_543_Update.img
Type: 58x
Version: 6.0.14.0543
Size (bytes): 32564

BP106A_MolocarH10S-7DBADDR_20250904_BL_103_APP_543_Update.img
Type: 58x
Version: 6.0.14.0543
Size (bytes): 32584

BT691_FeasycomFlash_20200910_BL_103_APP_103_Updat.img
Type: 58x
Version: 6.0.14.0103
Size (bytes): 31766

BT691_FeasycomFlash_20220914_BL_105_APP_404_Update.img
Type: 58x
Version: 6.0.14.404
Size (bytes): 33976

BT691_RCFIT_20250911_BL_105_APP_406_Update.img
Type: 58x
Version: 6.0.14.406
Size (bytes): 33940



SUOTA
Parameter settings

Device: Feasycom
File: BT691_FeasycomFlash_20220914_BL_105_APP_404_Update.img

Block size: 240
Image bank: OLDEST / BANK 1 / [✓ BANK 2]
Memory type: [✓ SPI] / I2C
MISO GPIO: P0_3
MOSI GPIO: P0_0
CS GPIO: P0_1
SCK GPIO: P0_4

SEND TO DEVICE

4

SUOTA
Uploading file

Uploading file BT691_FeasycomFlash_20220914_BL_105_APP_404_Update.img to device Feasycom.
Please wait until the process is completed...

33%
Sending chunk 96 of 283
Block 48 of 142

```
Block Speed: 5333 B/s
Current Speed: 6240 B/s
Average Speed: 6174 B/s
Max Block Speed: 8888 B/s
Min Block Speed: 2000 B/s

Upload size: 33977 bytes
Block size: 240 bytes
Chunk size: 182 bytes
Total blocks: 142
Total chunks: 283
Chunks per block: 2
Firmware CRC: 0x7f

Enable status notifications
Set memory device: 0x13000002
Set GPIO map: 0x03000104
Upload started
Set patch length: 240
```

5

SUOTA
Uploading file

Uploading file BT691_FeasycomFlash_20220914_BL_105_APP_404_Update.img to device Feasycom.
Please wait until the process is completed...

100%
Sending chunk 283 of 283
Block 142 of 142

Upload completed
Reboot device?
CANCEL OK

```
Elapsed Time: 5.559 seconds
Upload Time: 5.323 seconds
Average Speed: 6383 B/s
Max Block Speed: 9600 B/s
Min Block Speed: 2000 B/s

Upload size: 33977 bytes
Block size: 240 bytes
Chunk size: 182 bytes
Total blocks: 142
Total chunks: 283
Chunks per block: 2
Firmware CRC: 0x7f

Enable status notifications
```

CLOSE

6

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-BT691 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 The following parameter is 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

PDF Download