FC41D AT Commands Manual

Wi-Fi&Bluetooth Module Series

Version: 1.0

Date: 2022-09-27

Status: Released

At Quectel, our aim is to provide timely and comprehensive services to our customers. If you require any assistance, please contact our 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 offices. 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 us at: support@quectel.com.

Legal Notices

We offer information as a service to you. The provided information is based on your requirements and we make every effort to ensure its quality. You agree that you are responsible for using independent analysis and evaluation in designing intended products, and we provide reference designs for illustrative purposes only. Before using any hardware, software or service guided by this document, please read this notice carefully. Even though we employ commercially reasonable efforts to provide the best possible experience, you hereby acknowledge and agree that this document and related services hereunder are provided to you on an “as available” basis. We may revise or restate this document from time to time at our sole discretion without any prior notice to you.

Use and Disclosure Restrictions

License Agreements

Documents and information provided by us shall be kept confidential, unless specific permission is granted. They shall not be accessed or used for any purpose except as expressly provided herein.

Copyright

Our and third-party products hereunder may contain copyrighted material. Such copyrighted material shall not be copied, reproduced, distributed, merged, published, translated, or modified without prior written consent. We and the third party have exclusive rights over copyrighted material. No license shall be granted or conveyed under any patents, copyrights, trademarks, or service mark rights. To avoid ambiguities, purchasing in any form cannot be deemed as granting a license other than the normal non-exclusive, royalty-free license to use the material. We reserve the right to take legal action for noncompliance with abovementioned requirements, unauthorized use, or other illegal or malicious use of the material.

Trademarks

Except as otherwise set forth herein, nothing in this document shall be construed as conferring any rights to use any trademark, trade name or name, abbreviation, or counterfeit product thereof owned by Quectel or any third party in advertising, publicity, or other aspects.

Third-Party Rights

This document may refer to hardware, software and/or documentation owned by one or more third parties ("third-party materials"). Use of such third-party materials shall be governed by all restrictions and obligations applicable thereto.

We make no warranty or representation, either express or implied, regarding the third-party materials, including but not limited to any implied or statutory, warranties of merchantability or fitness for a particular purpose, quiet enjoyment, system integration, information accuracy, and non-infringement of any third-party intellectual property rights with regard to the licensed technology or use thereof. Nothing herein constitutes a representation or warranty by us to either develop, enhance, modify, distribute, market, sell, offer for sale, or otherwise maintain production of any our products or any other hardware, software, device, tool, information, or product. We moreover disclaim any and all warranties arising from the course of dealing or usage of trade.

Privacy Policy

To implement module functionality, certain device data are uploaded to Quectel's or third-party's servers, including carriers, chipset suppliers or customer-designated servers. Quectel, strictly abiding by the relevant laws and regulations, shall retain, use, disclose or otherwise process relevant data for the purpose of performing the service only or as permitted by applicable laws. Before data interaction with third parties, please be informed of their privacy and data security policy.

Disclaimer

a) We acknowledge no liability for any injury or damage arising from the reliance upon the information.
b) We shall bear no liability resulting from any inaccuracies or omissions, or from the use of the information contained herein.
c) While we have made every effort to ensure that the functions and features under development are free from errors, it is possible that they could contain errors, inaccuracies, and omissions. Unless otherwise provided by valid agreement, we make no warranties of any kind, either implied or express, and exclude all liability for any loss or damage suffered in connection with the use of features and functions under development, to the maximum extent permitted by law, regardless of whether such loss or damage may have been foreseeable.
d) We are not responsible for the accessibility, safety, accuracy, availability, legality, or completeness of information, advertising, commercial offers, products, services, and materials on third-party websites and third-party resources.

About the Document

Revision History

Version	Date	Author	Description
-	2021-06-18	Brave LIU/ Lidong ZHOU/	Creation of the document
1.0	2022-09-27	Brave LIU/ Lidong ZHOU/ Vic CHENG/ Lei WANG	First official release

Contents

Table Index

Figure Index

1 Introduction

This document outlines Wi-Fi, BLE, TCP/UDP, SSL, MQTT and HTTP(S)-related AT commands supported by Quectel FC41D module.

1.1. Definitions

- **<CR>** Carriage return character.
- **<LF>** Line feed character.
- **<...>** Parameter name. Angle brackets do not appear on the command line.
- **[...]** Optional parameter of a command or an optional part of TA information response. Square brackets do not appear on the command line. When an optional parameter is not given in a command, the new value equals its previous value or the default settings, unless otherwise specified.
- **Underline** Default setting of a parameter.

1.2. AT Command Syntax

All command lines must start with **AT** or **at** and end with **<CR>**. Information responses and result codes always start and end with a carriage return character and a line feed character: **<CR><LF><response><CR><LF>**. In tables presenting commands and responses throughout this document, only the commands and responses are presented, and **<CR>** and **<LF>** are deliberately omitted.

AT commands implemented by FC41D fall into two categories syntactically: “Basic” and “Extended”, as listed below:

- **Basic**

Basic command format is **AT<x><n>**, or **AT&<x><n>**, where **<x>** is the command, and **<n>** is/are the argument(s) of the command. For example, **ATE<n>** tells the DCE (Data Circuit-terminating Equipment) whether received characters should be echoed back to the DTE (Data Terminal Equipment) according to the value of **<n>**. **<n>** is optional and a default will be used if it is omitted.

- **Extended**

Extended commands can be executed in several modes, as shown in the following table:

Table 1: Types of AT Commands

Command Type	Syntax	Description
Test Command	**AT+<cmd>=?**	Test the existence of the corresponding command and return information about the type, value, or range of its parameter.
Read Command	**AT+<cmd>?**	Check the current parameter value of the corresponding command.
Write Command	**AT+<cmd>=<p1>[,<p2>[,<p3>[...]]]**	Set user-definable parameter value.
Execution Command	**AT+<cmd>**	Return a specific information parameter or perform a specific action.

Multiple commands can be placed on a single line using a semi-colon (**;**) between commands. In such cases, only the first command should have **AT** prefix. Commands can be in upper or lower case.

Spaces should be ignored when you enter AT commands, except in the following cases:

- Within quoted strings, where spaces are preserved;
- Within an unquoted string or numeric parameter;
- Within an IP address;
- Within the AT command name up to and including a **=**, **?** or **=?**.

On input, at least a carriage return is required. A newline character is ignored so it is permissible to use carriage return/line feed pairs on the input.

If no command is entered after the **AT** token, **OK** will be returned. If an invalid command is entered, **ERROR** will be returned.

Optional parameters, unless explicitly stated, need to be provided up to the last parameter being entered.

1.3. AT Command Responses

When the AT command processor has finished processing a line, it will output **OK**, **ERROR** or **+CME ERROR: <err>** to indicate that it is ready to accept a new command. Solicited information responses are sent before the final **OK**, **ERROR** or **+CME ERROR: <err>**.

Responses will be in the format of:

<CR><LF>+CMD1:<parameters><CR><LF>
<CR><LF>OK<CR><LF>

Or

<CR><LF><parameters><CR><LF>
<CR><LF>OK<CR><LF>

1.4. Declaration of AT Command Examples

The AT command examples in this document are provided to help you learn about the use of the AT commands introduced herein. The examples, however, should not be taken as Quectel's recommendations or suggestions about how to design a program flow or what status to set the module into. Sometimes multiple examples may be provided for one AT command. However, this does not mean that there is a correlation among these examples, or that they should be executed in a given sequence.

2 AT Commands Description

2.1. Description of Wi-Fi-Related AT Commands

2.1.1. AT+QRST Restart Module

This command restarts the module.

AT+QRST Restart Module	
Execution Command **AT+QRST**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

2.1.2. AT+QVERSION Get Firmware Version

This command gets firmware version of the module.

AT+QVERSION Get Firmware Version	
Execution Command **AT+QVERSION**	Response **+QVERSION: <version>** **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<version>	String type without double quotation marks. Firmware version number.

2.1.3. AT+QECHO Enable/Disable Echo Function

This command enables or disables echo function.

AT+QECHO Enable/Disable Echo Function	
Write Command **AT+QECHO=<enable>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configuration is saved automatically.

Parameter

<enable>	Integer type. Enable/disable echo function. 0 Disable 1 Enable

2.1.4. AT+QURCCFG Enable/Disable URC Report

This command enables or disables URC reports.

AT+QURCCFG Enable/Disable URC Report	
Write Command **AT+QURCCFG=<enable>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configuration is saved automatically.

Parameter

<enable>	Integer type. Enable/disable URC report. 0 Disable 1 Enable

2.1.5. AT+QPING Ping External IP

This command pings the external IP.

AT+QPING Ping External IP	
Write Command **AT+QPING=<IP>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<IP>	String type. External IP address.

2.1.6. AT+QGETIP Get IP Information

This command gets the module IP information.

AT+QGETIP Get IP Information	
Write Command **AT+QGETIP=<mode>**	Response **+QGETIP: <IP>,<gate>,<mask>,<DNS>** **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<mode>	String type without double quotation marks. Wi-Fi working mode. station STA Mode ap AP Mode
<IP>	String type without double quotation marks. Module IP address.
<gate>	String type without double quotation marks. Module gateway.
<mask>	String type without double quotation marks. Module gateway.
<DNS>	String type without double quotation marks. Module DNS address.

2.1.7. AT+QSETBAND Configure Serial Port Baud Rate

This command configures serial port baud rate.

AT+QSETBAND Configure Serial Port Baud Rate	
Write Command **AT+QSETBAND=<baud_rate>[,<save>]**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately.

Parameter

<baud_rate>	Integer type. Serial port baud rate.
<save>	Integer type. If it is omitted, the serial port baud rate will be saved. 0 Do not save 1 Save

2.1.8. AT+QWLANOTA Start OTA

This command starts firmware OTA.

AT+QWLANOTA Start OTA Upgrade	
Write Command **AT+QWLANOTA=<URL>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<URL>	String type without double quotation marks. Address where firmware package is stored on the server.

2.1.9. AT+QLOWPOWER Enter Low Power Mode

This command configures the module to enter low power mode.

AT+QLOWPOWER Enter Low Power Mode	
Execution Command **AT+QLOWPOWER**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

2.1.10. AT+QDEEPSLEEP Enter Deep Sleep Mode

This command configures the module to enter deep sleep mode.

AT+QDEEPSLEEP Enter Deep Sleep Mode	
Execution Command **AT+QDEEPSLEEP**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

2.1.11. AT+QWLMAC Get MAC Address

This command gets module MAC address.

AT+QWLMAC Get MAC Address	
Execution Command **AT+QWLMAC**	Response **+QWLMAC: <MAC>** **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<MAC>	MAC address of the module. Hexadecimal numbers separated by colons. Default value: c8:47:8c:42:00:48.

2.1.12. AT+QAIRKISS Enable/Disable AirKiss

This command enables or disables AirKiss function.

AT+QAIRKISS Enable/Disable AirKiss	
Write Command **AT+QAIRKISS=<enable>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<enable>	Integer type. Enable/Disable AirKiss function. 0 Disable 1 Enable

2.1.13. AT+QSTAST Query STA Mode State

This command queries STA mode state.

AT+QSTAST Query STA Mode State	
Execution Command **AT+QSTAST**	Response **+QSTAST: <state>** **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<state>	String type without double quotation marks. STA mode state.

	STATION_DOWN	Disabled
	STATION_UP	Enabled

2.1.14. AT+QSTADHCP Enable/Disable DHCP Service in STA Mode

This command enables or disables DHCP service in STA mode.

AT+QSTADHCP Enable/Disable DHCP Service in STA Mode	
Write Command **AT+QSTADHCP=<enable>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configuration is not saved.

Parameter

<enable>	Integer type. Enable/disable DHCP service in STA mode. 0 Disable 1 Enable

2.1.15. AT+QSTADHCPDEF Enable/Disable DHCP Service in STA Mode and Save Configuration

This command enables or disables DHCP service in STA mode and saves the configuration.

AT+QSTADHCPDEF Enable/Disable DHCP Service in STA Mode and Save Configuration	
Write Command **AT+QSTADHCPDEF=<enable>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configuration is saved automatically.

Parameter

<enable>	Integer type. Enable/disable DHCP service in STA mode and save the

	configuration. 0 Disable. 1 Enable

2.1.16. AT+QSTASTATIC Configure Static IP of STA Mode

This command configures static IP of STA mode.

AT+QSTASTATIC Configure Static IP of STA Mode	
Write Command **AT+QSTASTATIC=<IP>,<mask>,<gate>,<DNS>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<IP>	String type without double quotation marks. Static IP address of STA mode.
<mask>	String type without double quotation marks. Module subnet mask.
<gate>	String type without double quotation marks. Module gateway.
<DNS>	String type without double quotation marks. Module DNS address.

2.1.17. AT+QSTASTOP Disable STA Mode

This command disables STA mode.

AT+QSTASTOP Disable STA Mode	
Execution Command **AT+QSTASTOP**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

2.1.18. AT+QSOFTAP Enable AP Mode

This command enables AP mode.

AT+QSOFTAP Enable AP Mode	
Write Command **AT+QSOFTAP=<SSID>[,<key>]**	Response **OK** Or **ERROR**
Maximum Response Time	3300 ms (encrypted hotspot enabled)/300 ms (passwordless hotspot enabled)
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<SSID>	String type without double quotation marks. AP name. Range: 1–32. Unit: byte.
<key>	String type without double quotation marks. AP security key. Range: 8–63. Unit: byte. If it is omitted, the passwordless hotspot will be enabled.

2.1.19. AT+QAPSTATE Query AP Mode State

This command queries AP mode state.

AT+QAPSTATE Query AP Mode State	
Execution Command **AT+QAPSTATE**	Response **+QAPSTATE: <state>** **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<state>	String type without double quotation marks. AP mode state.	
	SOFTAP_DOWN	Disabled
	SOFTAP_UP	Enabled

2.1.20. AT+QAPSTATIC Configure Static IP of AP Mode

This command configures static IP of AP mode.

AT+QAPSTATIC Configure Static IP of AP Mode	
Write Command **AT+QAPSTATIC=<IP>,<mask>,<gate>,<DNS>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<IP>	String type without double quotation mark. Static IP address of AP mode.
<mask>	String type without double quotation mark. Module gateway.
<gate>	String type without double quotation mark. Module gateway.
<DNS>	String type without double quotation mark. Module DNS address.

2.1.21. AT+QSOFTAPSTOP Disable AP Mode

This command disables AP mode.

AT+QSOFTAPSTOP Disable AP Mode	
Execution Command **AT+QSOFTAPSTOP**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

2.1.22. AT+QSTAAPINFO Connect an AP Hotspot

This command connects an AP hotspot to enable STA mode.

AT+QSTAAPINFO Connect an AP Hotspot	
Write Command **AT+QSTAAPINFO=<SSID>[,<pwd>]**	Response **OK** Or

	ERROR
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<SSID>	String type without double quotation marks. Name of AP hotspot to be connected. Range: 1–32. Unit: byte.
<pwd>	String type without double quotation marks. Security key of AP hotspot to be connected. Range: 8–63. Unit: byte. If it is omitted, the passwordless hotspot will be connected.

2.1.23. AT+QSTAAPINFODEF Connect a Hotspot and Save Hotspot Information

This command connects a hotspot to enable STA mode and saves the connected hotspot information.

AT+QSTAAPINFODEF Connect a Hotspot and Save Hotspot Information	
Write Command **AT+QSTAAPINFODEF=<SSID>[,<pwd>]**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configurations are saved automatically.

Parameter

<SSID>	String type without double quotation marks. Name of AP hotspot to be connected. Range: 1–32. Unit: byte.
<pwd>	String type without double quotation marks. Security key of AP hotspot to be connected. Range: 8–63. Unit: byte. If it is omitted, the passwordless hotspot will be enabled.

2.1.24. AT+QGETWIFISTATE Query Connected Hotspot

This command queries the connected hotspot when the module is in STA mode.

AT+QGETWIFISTATE Query Connected Hotspot	
Execution Command **AT+QGETWIFISTATE**	Response **+QGETWIFISTATE: ssid=<SSID>,bssid=<BSSID>,rssi=<RSSI>,channel=<channel>,cipher_type=<cipher_type>** **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<SSID>	String type. Name of connected Wi-Fi hotspot.
<BSSID>	String type without double quotation marks. MAC address of Wi-Fi network card.
<RSSI>	Integer type. Wi-Fi signal strength.
<channel>	Integer type. Channel used by module to connect and communicate with Wi-Fi hotspot.
<cipher_type>	String type without double quotation marks. Encryption type.

2.1.25. AT+QWSCAN Query Scanned Hotspot Information

This command queries the scanned hotspot information.

AT+QWSCAN Query Scanned Hotspot Information	
Execution Command **AT+QWSCAN**	Response **+QWSCAN: <SSID>,<PSK_type>,<RSSI>,<BSSID>,<channel>** **[…]** **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<SSID>	String type. Name of scanned Wi-Fi hotspot.
<PSK_type>	String type without double quotation marks. Encryption type.
<RSSI>	Integer type. Wi-Fi signal strength.
<BSSID>	String type without double quotation marks. MAC address of Wi-Fi network card.
<channel>	Integer type. Channel used by module to connect and communicate with Wi-Fi hotspot.

2.1.26. AT+QWEBCFG Enable/Disable Configuring Wi-Fi via Web

This function enables or disables configuring Wi-Fi via Web.

AT+QWEBCFG Enable/Disable Configuring Wi-Fi via Web	
Write Command **AT+QWEBCFG=<enable>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configuration is saved automatically.

Parameter

<enable>	Integer type. Enable/disable configuring Wi-Fi via Web. 0 Disable 1 Enable

2.2. Description of BLE-Related AT Commands

2.2.1. AT+QBLEINIT Initialize BLE Service

This command initializes BLE service.

AT+QBLEINIT Initialize BLE Service	
Write Command **AT+QBLEINIT=<role>**	Response **OK** Or **ERROR**
Read Command **AT+QBLEINIT?**	Response

	+QBLEINIT:<role> **OK** or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<role>	Integer type. Initialize BLE service. 1 Module is operating as a central device for initializing BLE. 2 Module is operating as a peripheral device for initializing BLE. 3 Module is operating as a peripheral device for configuring Wi-Fi via BLE.

2.2.2. AT+QBLEADDR Query BLE Device Address

This command queries BLE device address.

AT+QBLEADDR Query BLE Device Address	
Read Command **AT+QBLEADDR?**	Response **+QBLEADDR:<BLE_addr>** **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<BLE_addr>	String type without double quotation marks. BLE device address. A 48-bit address is represented in a string of hexadecimal numbers, such as 58:D3:91:01:02:03.

2.2.3. AT+QBLENAME Set BLE Name

This command sets a BLE name.

AT+QBLENAME Set BLE Name	
Read Command **AT+QBLENAME?**	Response **+QBLENAME: <BLE_name>** **OK** Or **ERROR**
Write Command **AT+QBLENAME=<BLE_name>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<BLE_name>	String type without double quotation marks. BLE name. Maximum length: 25 bytes.

2.2.4. AT+QBLEADVPARAM Configure BLE Advertising Parameters

This command configures BLE advertising parameters when the module is operating as a peripheral device.

AT+QBLEADVPARAM Configure BLE Advertising Parameters	
Write Command **AT+QBLEQADVPARAM=<adv_int_min>,<adv_int_max>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<adv_int_min>	Integer type. Minimum advertising interval for undirected advertising and low duty cycle directed advertising. Range: 0x0020–0x4000 (Time range: 20 ms–

<adv_int_max>	10.24 s). Default value: 0x0800 (1.28 s). Unit: timeslot (1 timeslot = 0.625 ms). Integer type. Maximum advertising interval for undirected advertising and low duty cycle directed advertising. Range: 0x0020–0x4000 (Time range: 20 ms–10.24 s). Default value: 0x0800 (1.28 s). Unit: timeslot (1 timeslot = 0.625 ms).

NOTE

To configure advertising parameters, **AT+QBLEADVPARAM** should be executed before initiating advertising with **AT+QBLEADVSTART**.

2.2.5. AT+QBLEADVDATA Set BLE Advertising Data

This command sets BLE advertising data when the module is operating as a peripheral device.

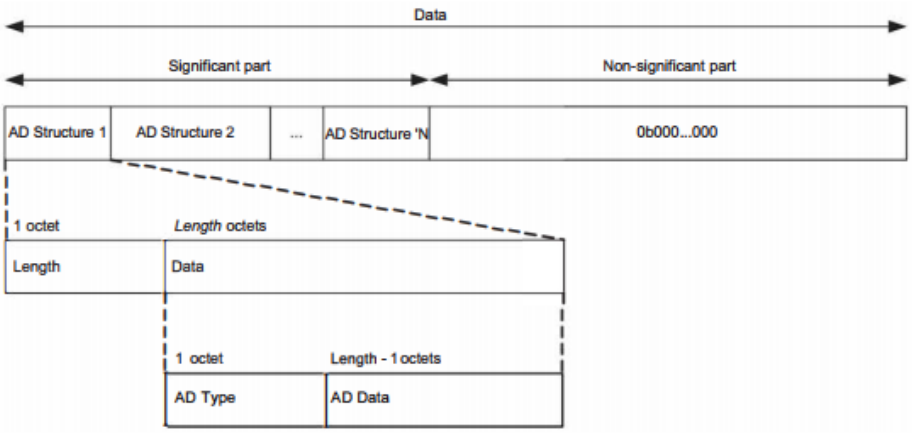

Figure 1: Advertising and Scan Response Data Format

AT+QBLEADVDATA Set BLE Advertising Data	
Write Command **AT+QBLEADVDATA=<adv_data>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configuration is not saved.

Parameter

<adv_data>	String type without double quotation marks. Advertising data (AD). It consists

	of three fields (i.e., multiple AD Structures). The composition conforms to the message format shown above and the content must be a hexadecimal string. Length Length of AD structure. The length includes AD type and AD data but not the length of the field which is 1 byte long. The maximum length is 0x1e, i.e., the maximum length of a data field is 30 bytes. AD Type Advertising data type, such as TX Power Level (0x0A), Local Name (0x09), Le Role (0x1C) and Service UUIDs (0x16). After the peer scans the advertisement, the meaning of the advertising data can be determined from the AD Type. AD Data Advertising data in big-endian byte order.

NOTE

For details of AD types, see *Core Specification 5.2* (https://www.bluetooth.com/specifications/specs/core-specification/).

2.2.6. AT+QBLEGATTSSRV Establish a BLE Service

This command establishes a BLE service when the module is operating as a peripheral device.

AT+QBLEGATTSSRV Establish a BLE Service	
Write Command **AT+QBLEGATTSSRV=<srv_UUID>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configuration is not saved.

Parameter

<srv_UUID>	String type without double quotation marks. BLE service UUID. Length: 2 bytes or 16 bytes.

NOTE

Only one BLE service is established at a time with this command.

2.2.7. AT+QBLEGATTSCHAR Set BLE Characteristic UUID

This command sets BLE characteristic UUID when the module is operating as a peripheral device.

AT+QBLEGATTSCHAR Set BLE Characteristic UUID	
Write Command **AT+QBLEGATTSCHAR=<char_UUID>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configuration is not saved.

Parameter

<char_UUID>	String type without double quotation marks. Characteristic UUID. Length: 2 bytes or 16 bytes.

2.2.8. AT+QBLEADVSTART Start BLE Advertising

This command starts BLE advertising when the module is operating as a peripheral device.

AT+QBLEADVSTART Start BLE Advertising	
Execution Command **AT+QBLEADVSTART**	Response **OK** Or **ERROR**
Maximum Response Time	1000 ms
Characteristics	/

2.2.9. AT+QBLEADVSTOP Stop BLE Advertising

This command stops BLE advertising when the module is operating as a peripheral device.

AT+QBLEADVSTOP Stop BLE Advertising	
Execution Command **AT+QBLEADVSTOP**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms

Characteristics	/

2.2.10. AT+QBLEGATTSNTFY Send GATT Data

This command sends GATT data when the module is operating as a peripheral device.

AT+QBLEGATTSNTFY Send GATT Data	
Write Command **AT+QBLEGATTSNTFY=<UUID>[,<hex_length>],<data>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<UUID>	String type without double quotation marks. Characteristic UUID. Length: 2 bytes or 16 bytes.
<hex_length>	Integer type. Hex data length. The module will convert **<data>** to hexadecimal format before sending data when this optional parameter is used. For example, if **<data>** is 123456, the module will convert the data to 0x123456 and send it.
<data>	String type without double quotation marks. GATT data.

2.2.11. AT+QBLESCAN Start/Stop BLE Scan

This command starts or stops BLE scan when the module is operating as a central device.

AT+QBLESCAN Start/Stop BLE Scan	
Write Command **AT+QBLESCAN=<scan>[,<timeout>]**	Response If **<scan>** is 0: **OK** Or **ERROR** If **<scan>** is 1: **OK** **+QBLESCAN:<name>,<address_type>,<address>** Or **ERROR**

	If **<scan>** is 2: **OK** **+QBLESCAN:<name>,<address_type>,<address>,<adv_data>** Or **ERROR**
Maximum Response Time	1000 ms
Characteristics	/

Parameter

<scan>	Integer type. Start/stop BLE scan. 0 Stop 1 Start BLE scan but do not output advertising data or scan response data. 2 Start BLE scan and output advertising data and scan response data. It is not required to manually stop BLE scan in this mode.
<timeout>	Integer type. Scan duration. Range: 500–10000. Unit: ms. It must be set when **<scan>** is 2.
<name>	String type without double quotation marks. BLE device name.
<address_type>	Integer type. BLE device address type. 0 Public address 1 Random address
<address>	String type without double quotation marks. BLE device address. Length: 12 bytes.
<adv_data>	String type without double quotation marks. Scanned BLE data and scan response data.

2.2.12. AT+QBLESCANPARAM Set BLE Scan Parameters

This command sets BLE scan parameters when the module is operating as a central device.

AT+QBLESCANPARAM Set BLE Scan Parameters	
Write Command **AT+QBLESCANPARAM=<scan_interval>,<scan_window>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<scan_interval>	Integer type. LE scan interval. Range: 0x0012–0x1000 (Time range: 11.25 ms to 2.56 s). Default value: 0x0064. Unit: timeslot (1 timeslot = 0.625 ms).
<scan_window>	Integer type. LE scan duration. **<scan_window>** shall be less than or equal to **<scan_interval>**. Range: 0x0011–0x1000 (Time range: 10.625 ms–2.56 s). Default value: 0x001E. Unit: timeslot (1 timeslot = 0.625 ms).

2.2.13. AT+QBLECONN Connect a Peripheral Device

This command connects a peripheral when the module is the central device.

AT+QBLECONN Connect a Peripheral Device	
Write Command **AT+QBLECONN=<addr_type>,<peer_addr>**	Response **OK** Or **ERROR**
Maximum Response Time	3000 ms
Characteristics	The command takes effect immediately. The configurations are not saved. The connection timeout is 18 seconds.

Parameter

<addr_type>	Integer type. Peripheral device address type. 0 Public address 1 Random address
<peer_addr>	String type without double quotation marks. Peripheral device address.

2.2.14. AT+QBLECONNPARAM Configure Connection Parameters

This command configures connection parameters when the module is the central device.

AT+QBLECONNPARAM Configure Connection Parameters	
Write Command **AT+QBLECONNPARAM=<con_interval>,<timeout>,<latency>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<con_interval>	Integer type. Connection interval. Range: 0x0006–0x0C80 (Time range: 7.5 ms to 4 s). Default value: 30. Unit: 1.25 ms.
<timeout>	Integer type. LE Link supervision timeout. Range: 0x000A to 0x0C80 (Time range: 100 ms to 32 s). Default value: 500. Unit: 10 ms.
<latency>	Integer type. Slave latency for the connection in a certain number of connection events. Range: 0–499 (0x0000 to 0x01F3) Default value: 0.

2.2.15. AT+QBLECFGMTU Configure Maximum Transmission Unit for BLE

This command configures maximum transmission unit for BLE when the module is the central device.

AT+QBLECFGMTU Configure Maximum Transmission Unit for BLE	
Read Command **AT+QBLECFGMTU?**	Response **+QBLECFGMTU: <MTU_value>** **OK** Or **ERROR**
Write Command **AT+QBLECFGMTU=<MTU_value>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configuration is not saved.

Parameter

<MTU_value>	Integer type. Maximum transmission unit value. Range: 23–512. Unit: byte.

2.2.16. AT+QBLEGATTCNTFCFG Turn on/off Notification

This command enables or disables notification when the module is operating as central device.

AT+QBLEGATTCNTFCFG Turn on/off Notification	
Write Command **AT+QBLEGATTNTFCFG=<UUID>,<enable>**	Response **OK** Or **ERROR**

Maximum Response Time	300 ms
Characteristics	/

Parameter

<UUID>	String type without double quotation marks. Characteristic UUID. Length: 2 bytes or 16 bytes.
<enable>	Integer type. Enables/Disables notification. 0 Disable 1 Enable

2.2.17. AT+QBLEGATTCWR Send Data

This command sends GATT data when the module is operating as central device.

AT+QBLEGATTCWR Send Data	
Write Command **AT+QBLEGATTCWR=<UUID>[,<hex_length>],<data>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<UUID>	String type without double quotation marks. Characteristic UUID. Length: 2 bytes or 16 bytes.
<hex_length>	Integer type. Hex data length. When this optional parameter is applied, the module converts **<data>** to hexadecimal format before sending it. For example, if **<data>** is 123456, the module converts the data to 0x123456 and then sends it.
<data>	String type without double quotation marks. GATT data.

2.2.18. AT+QBLEGATTCRD Read Data

This command reads GATT data when the module is the central device.

AT+QBLEGATTCRD Read Data	
Write Command **AT+QBLEGATTCRD=<UUID>**	Response **OK**

	<data> Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configuration is not saved.

Parameter

<UUID>	String type without double quotation marks. Characteristic UUID. Length: 2 bytes or 16 bytes.
<data>	String type without double quotation marks. Read GATT data.

2.2.19. AT+QBLEDISCONN Disconnect BLE Connection

This command disconnects a BLE connection.

AT+QBLEDISCONN Disconnect BLE Connection	
Execution Command **AT+QBLEDISCONN**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

2.2.20. AT+QBLESTAT Query BLE Device State

This command queries the state of BLE device.

AT+QBLESTAT Query BLE Device State	
Execution Command **AT+QBLESTAT**	Response **+QBLESTAT: <BLE_state>** **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<BLE_state>	String type without double quotation marks. Current state of BLE device.	
	NOINIT	Uninitialized
	INIT	Initialized
	ADVERTISING	Advertising
	NOADVERTISING	Not advertising
	CONNECTED	Connected
	DISCONNECTED	Disconnected

2.3. Description of TCP/UDP Related AT commands

2.3.1. AT+QICFG Configure Optional Parameters for TCP/UDP Socket Service

AT+QICFG Configure Optional Parameters for TCP/UDP Socket Service	
Test Command **AT+QICFG=?**	Response **+QICFG: "transpktsize",(**range of supported **<transpktsize>**s**)** **+QICFG: "transwaittm",(**range of supported **<transwaittm>**s**)** **+QICFG: "accept/mode",(**list of supported **<accept_mode>**s**)** **+QICFG: "tcp/tw_cycle"**(list of supported **<tcp_tw_cycle>**s**)** **+QICFG: "datatype",(**list of supported **<data_type>**s**)** **OK**
Write Command Configure sub-package length of data to be sent in transparent transmission mode **AT+QICFG="transpktsize"[,<transpktsize>]**	Response If the optional parameter is omitted, query the current configuration: **+QICFG: "transpktsize",<transpktsize>** **OK** If the optional parameter is specified, configure sub-package length of data to be sent in transparent transmission mode: **OK** Or **ERROR**
Write Command Configure user data reception waiting time of serial port in transparent transmission mode **AT+QICFG="transwaittm"[,<transwaittm>]**	Response If the optional parameter is omitted, query the current configuration: **+QICFG: "transwaittm",<transwaittm>** **OK**

	If the optional parameter is specified, configure user data reception waiting time of serial port in transparent transmission mode: **OK** Or **ERROR**
Write Command Configure acceptance mode of "TCP Incoming" **AT+QICFG="accept/mode"[,<accept_mode>]**	Response If the optional parameter is omitted, query the current configuration: **+QICFG: "accept/mode",<accept_mode>** **OK** If the optional parameter is specified, configure acceptance mode of "TCP Incoming": **OK** Or **ERROR**
Write Command Configure release method of port occupied by "TCP" connection **AT+QICFG="tcp/tw_cycle"[,<tcp_tw_cycle>]**	Response If the parameter is omitted, query the current configuration: **+QICFG: "tcp/tw_cycle",<tcp_tw_cycle>** **OK** If the parameter is specified, configure the release method of port occupied by "TCP" connection: **OK** Or **ERROR**
Write Command Configure type of data to be sent **AT+QICFG="datatype"[,<data_type>]**	Response If the optional parameter is omitted, query the current configuration: **+QICFG: "datatype",<data_type>** **OK** If the optional parameter is specified, configure type of data to be sent: **OK** Or **ERROR**
Maximum Response Time	300 ms

Characteristics	/

Parameter

<transpktsize>	Integer type. Sub-package length of data to be sent. Range: 1–1460. Default value: 1024. Unit: byte.
<transwaittm>	Integer type. Data reception waiting time if the received data is less then **<transpktsize>** in transparent transmission mode. If **<transwaittm>** is 0, the data is sent as soon as it is received. Range: 0–20. Default value: 2. Unit: 100 ms.
<accept_mode>	Integer type. Acceptance mode of TCP incoming connection. 0 Accept automatically 1 Accept manually with **AT+QIACCEPT**
<tcp_tw_cycle>	Integer type. Release method of port occupied by "TCP" connection. 0 Release immediately 1 Release delay
<data_type>	Integer type. Type of data to be sent. 0 Non-hex string 1 Hex string

2.3.2. AT+QIOPEN Open TCP/UDP Socket Service

This command opens a TCP/UDP socket service. The maximum response time for establishing a TCP connection as a client is determined by the timeout of the TCP three-way handshake. The timeout of the TCP three-way handshake supported by FC41D module is 120 seconds. When **<service_type>** is "UDP "/"TCP LISTENER"/"UDP SERVICE", the response timeout is about 300 milliseconds.

AT+QIOPEN Open TCP/UDP Socket Service	
Test Command **AT+QIOPEN=?**	Response **+QIOPEN: (**range of supported **<socketID>**s**),(**list of supported **<service_type>**s**),"ip/dns",(**range of supported **<remote_port>**s**),(**range of supported **<local_port>**s**),(**range of supported **<access_mode>**s**)** **OK**
Write Command **AT+QIOPEN=<socketID>,<service_type>,<remoteIP/domain_name>,<remote_port>,<local_port>,<access_mode>**	Response If **<access_mode>** is 2: **CONNECT** Or **ERROR** If **<access_mode>** is not 2: **OK**

	+QIOPEN: <socketID>,<err> Or **ERROR**
Maximum Response Time	/
Characteristics	/

Parameter

<socketID>	Integer type. Socket ID. Range: 0–11.
<service_type>	String type. Socket service type. "TCP" Start a TCP connection as a client "UDP" Start a UDP connection as a client "TCP LISTENER" Start a TCP server to listen for incoming TCP connections "UDP SERVICE" Start a UDP service
<remoteIP>	String type. IP address of the remote server. It is valid only when **<service_type>** is "TCP "/"UDP".
<domain_name>	String type. Remote server domain name. It is valid only when **<service_type>** is "TCP"/"UDP".
<remote_port>	Integer type. Remote server port. It is valid only when **<service_type>** is "TCP"/"UDP". Range: 1–65535.
<local_port>	Integer type. Local port. When **<service_type>** is "TCP LISTENER"/"UDP SERVICE", local port must be specified. Range: 1–65535.
<access_mode>	Integer type. Data access mode of serial port. 0 Buffer access mode. Socket sends and receives data with AT commands. 1 Direct push Mode. Socket sends data in AT command and receives data in URC format. 2 Transparent transmission mode. Serial port is exclusively used for sending/receiving data directly to/from the Internet.
<err>	Integer type. Result code. See ***Chapter 5*** for details.

NOTE

When **<service_type>** is "TCP LISTENER"/"UDP SERVICE", **<access_mode>** cannot be set to 2.

2.3.3. AT+QISTATE Query State of TCP/UDP Socket Service

AT+QISTATE Query State of TCP/UDP Socket Service	
Write Command **AT+QISTATE=<socketID>**	Response **+QISTATE: <socketID>,<service_type>,<remote_addr>,<remote_port>,<local_port>,<state>** **OK**
Read Command **AT+QISTATE?**	Response Return the state of all existing connections: **[+QISTATE: <socketID>,<service_type>,<remote_addr>,<remote_port>,<local_port>,<state>]** **[…]** **OK**
Execution Command **AT+QISTATE**	Response Return the state of all existing connections: **[+QISTATE: <socketID>,<service_type>,<remote_addr>,<remote_port>,<local_port>,<state>]** **[…]** **OK**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<socketID>	Integer type. Socket ID. Range: 0–11.
<service_type>	String type. Socket service type. "TCP" Start a TCP connection as a client "UDP" Start a UDP connection as a client "TCP LISTENER" Start a TCP server to listen for incoming TCP connections "UDP SERVICE" Start a UDP service "TCP INCOMING" Start a TCP connection accepted by the TCP server
<remote_addr>	String type. IP address of the remote server. It is valid only when **<service_type>** is "TCP"/"UDP"/"TCP INCOMING".
<remote_port>	Integer type. Remote server port. It is valid only when **<service_type>** is "TCP"/"UDP"/"TCP INCOMING".
<local_port>	Integer type. Local port.
<state>	Integer type. Socket service state. 0 "Initial" client connection is not established

1	"Opening" client is connecting or server is trying to listen
2	"Connected" client connection is established
3	"Listening" server is listening
4	"Closing" client connection is closing

2.3.4. AT+QISEND Send Data Through TCP/UDP Socket Service

AT+QISEND Send Data Through TCP/UDP Socket Service	
Test Command **AT+QISEND=?**	Response **+QISEND: (**range of supported **<socketID>**s**),(**range of supported **<send_len>**s**),"data"[,"remote_ip",(**range of supported **<remote_port>**s**)]** **OK**
Write Command When **<send_len>** is 0, query the sent data **AT+QISEND=<socketID>,0**	Response **+QISEND: <total_send_size>,<acked_size>,<unack_size>** **OK** Or **ERROR**
Write Command **<service_type>** is "UDP SERVICE" **AT+QISEND=<socketID>,<send_len>,<data>,<remote_ip>,<remote_port>**	Response **+QISEND: <actual_send_len>** **OK** Or **ERROR**
Write Command **<service_type>** is "TCP"/"UDP"/"TCP INCOMING" **AT+QISEND=<socketID>,<send_len>,<data>**	Response **+QISEND: <actual_send_len>** **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<socketID>	Integer type. Socket ID. Range: 0–11.
<send_len>	Integer type. Length of data to be sent. Range: 0–1460. Unit: byte.
<data>	String type. Data to be sent in hex string by default.

<remote_ip>	String type. Destination address of data to be sent. It is valid only when **<service_type>** is "UDP SERVICE".
<remote_port>	Integer type. Destination port of data to be sent. It is valid only when **<service_type>** is "UDP SERVICE". Range: 1–65535.
<actual_send_len>	Integer type. Actual length of data written to socket. Unit: byte.
<total_send_size>	Integer type. Total length of data written to socket. Unit: byte.
<acked_size>	Integer type. Length of acknowledged data. Unit: byte.
<unacked_size>	Integer type. Length of unacknowledged data. Unit: byte.

2.3.5. AT+QIRD Read Data Received from TCP/UDP Socket Service

AT+QIRD Read Data Received from TCP/UDP Socket Service	
Test Command **AT+QIRD=?**	Response **+QIRD: (**range of supported **<socketID>**s**),(**range of supported **<read_len>**s**)** **OK**
Write Command When **<send_len>** is 0, query the received data **AT+QIRD=<socketID>,0**	Response **+QISEND: <total_recv_size>,<read_size>,<unread_size>** **OK** Or **ERROR**
Write Command **<service_type>** is "UDP SERVICE" **AT+QIRD=<socketID>,<read_len>**	Response **+QIRD: <actual_read_len>,<remote_addr>,<remote_port>** **<data>** **OK** Or **ERROR**
Write Command **<service_type>** is "TCP"/"UDP"/"TCP INCOMING" **AT+QIRD=<socketID>,<read_len>**	Response **+QISEND: <actual_read_len>** **<data>** **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<socketID>	Integer type. Socket ID. Range: 0–11.
<read_len>	Integer type. Length of data to be read. Range: 0–1500. Unit: byte.
<total_recv_size>	Integer type. Total length of data received by socket. Unit: byte.
<read_size>	Integer type. Length of read data. Unit: byte.
<unread_size>	Integer type. Length of unread data. Unit: byte.
<actual_read_len>	Integer type. Length of actually read data. Unit: byte.
<remote_addr>	String type. Source address of received data. It is valid only when **<service_type>** is "UDP SERVICE".
<remote_port>	Integer type. Source port of received data. It is valid only when **<service_type>** is "UDP SERVICE".
<data>	String type without double quotation marks. Actually read data. Unit: byte.

2.3.6. AT+QIACCEPT Accept/Reject Remote Incoming Connection Request from TCP/UDP Socket Service

AT+QIACCEPT Accept/Reject Remote Incoming Connection Request from TCP/UDP Socket Service	
Test Command **AT+QIACCEPT=?**	Response **+QIACCEPT: (**range of supported **<listener_socketID>**s**),(**list of supported **<accept>**s**),(**range of supported **<incoming_socketID>**s**)** **OK**
Write Command Accept/Reject incoming connection request **AT+QIACCEPT=<listener_socketID>,<accept>[,<incoming_socketID>]**	Response **[+QIACCEPT: <incoming_socketID>,<remote_addr>,<remote_port>]** **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<listener_socketID>	Integer type. Socket ID of TCP server. Range: 0–11.
<accept>	Integer type. Accept or reject remote incoming connection request of TCP/UDP socket service.

	0 Reject 1 Accept
<incoming_socketID>	Integer type. Socket ID for incoming connection. It is valid only when **<accept>** is 1. Range: 0–11.
<remote_addr>	String type. Source address of incoming connection.
<remote_port>	Integer type. Source port of incoming connection.

2.3.7. AT+QISWTMD Switch Data Access Mode

AT+QISWTMD Switch Data Access Mode	
Test Command **AT+QISWTMD=?**	Response **+QISWTMD: (**range of supported **<socketID>**s**),(**range of supported **<access_mode>**s**)** **OK**
Write Command **AT+QISWTMD=<socketID>,<access_mode>**	Response **OK** Or **CONNECT** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<socketID>	Integer type. Socket ID. Range: 0–11.
<access_mode>	Integer type. Data access mode of serial port. 0 Buffer access mode. Socket sends and receives data with AT commands. 1 Direct push Mode. Socket sends data in AT command and receives data in URC format. 2 Transparent transmission mode. Serial port is exclusively used for sending/receiving data directly to/from the Internet.

2.3.8. AT+QICLOSE Close TCP/UDP Socket Service

AT+QICLOSE Close TCP/UDP Socket Service	
Test Command **AT+QICLOSE=?**	Response **+QICLOSE: (**range of supported **<socketID>**s**),(**range of supported **<close_timeout>**s**)** **OK**
Write Command **AT+QICLOSE=<socketID>[,<close_timeout>]**	Response **OK** **+QIURC: "closed",<socketID>** Or **ERROR**
Maximum Response Time	/
Characteristics	/

Parameter

<socketID>	Integer type. Socket ID. Range: 0–11.
<close_timeout>	Integer type. Timeout for closing TCP/UDP socket service. Range: 1–60. Default value: 2. Unit: second.

2.3.9. AT+QIGETERROR Query Result Code Related to TCP/UDP Socket Service

AT+QIGETERROR Query Result Code Related to TCP/UDP Socket Service	
Execution Command **AT+QIGETERROR**	Response **+QIGETERROR: <err>,<description>** **OK**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<err>	Integer type. Result code. See ***Chapter 5*** for details.
<description>	String type. Result code description.

2.3.10. ATO Enter Transparent Transmission Mode

ATO Enter Transparent Transmission Mode	
Execution Command **ATO**	Response **CONNECT** Or **NO CARRIER**
Maximum Response Time	300 ms
Characteristic	/

NOTE

If the socket connection has not been established before, **ATO** returns **NO CARRIER**.

2.3.11. +++ Exit Transparent Transmission Mode

+++ Exit Transparent Transmission Mode	
Execution Command **+++**	Response **OK**
Maximum Response Time	300 ms
Characteristics	/

NOTE

After exiting transparent transmission mode with **+++**, if the socket connection is active , the connection enters transparent transmission mode again with **ATO**.

2.4. Description of SSL-Related AT Commands

2.4.1. AT+QSSLCFG Configure SSL Context Parameters

AT+QSSLCFG Configure SSL Context Parameters	
Test Command **AT+QSSLCFG=?**	Response **+QSSLCFG: "version",(**range of supported **<SSL_ctxID>**s**),(**range of supported **<SSL_version>**s**)** **+QSSLCFG: "verify",(**range of supported **<SSL_ctxID>**s**),(**range of supported **<verify_level>**s**)** **+QSSLCFG: "ciphersuite",(**range of supported **<SSL_ctxID>**s**),<cs_id>** **+QSSLCFG: "negotiatetimeout",(**range of supported **<SSL_ctxID>**s**),(**range of supported **<negotiate_time>**s**)** **+QSSLCFG: "sni",(**range of supported **<SSL_ctxID>**s**),(**range of supported **<SNI_value>**s**)** **OK**
Write Command Configure SSL version **AT+QSSLCFG="version",<SSL_ctxID>[,<SSL_version>]**	Response If the optional parameter is omitted, query the current configuration: **+QSSLCFG: "version",<SSL_ctxID>,<SSL_version>** **OK** If the optional parameter is specified, configure the SSL version: **OK** Or **ERROR**
Write Command Configure SSL verification level **AT+QSSLCFG="verify",<SSL_ctxID>[,<verify_level>]**	Response If the optional parameter is omitted, query the current configuration: **+QSSLCFG: "verify",<SSL_ctxID>,<verify_level>** **OK** If the optional parameter is specified, configure the SSL verification level: **OK** Or **ERROR**
Write Command	Response

Configure SSL cipher suite **AT+QSSLCFG="ciphersuite",<SSL_ctxID>[,[<cs_id>[,<cs_id>[,…]]]**	If the optional parameters are omitted, query the current configuration: **+QSSLCFG: "ciphersuite",<SSL_ctxID>[,<cs_id>[,<cs_id>[,…]]]** **OK** If any of the optional parameters is specified, configure the SSL cipher suite: **OK** Or **ERROR**
Write Command Configure SSL negotiation timeout **AT+QSSLCFG="negotiatetimeout",<SSL_ctxID>[,<negotiate_time>]**	Response If the optional parameter is omitted, query the current configuration: **+QSSLCFG: "negotiatetimeout",<SSL_ctxID>,<negotiate_time>** **OK** If the optional parameter is specified, configure SSL negotiation timeout: **OK** Or **ERROR**
Write Command Enable or disable TLS server name indication **AT+QSSLCFG="sni",<SSL_ctxID>[,<SNI_value>]**	Response If the optional parameter is omitted, query the current configuration: **+QSSLCFG: "sni",<SSL_ctxID>,<SNI_value>** **OK** If the optional parameter is specified, enable or disable TLS server name indication: **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<SSL_ctxID>	Integer type. SSL context ID. Range: 0–5.
<SSL_version>	Integer type. SSL version. 0 SSL3.0. 1 TLS1.0 2 TLS1.1 3 TLS1.2 4 All
<verify_level>	Integer type. SSL verification level. 0 No verification 1 One-way verification (Client verifies server legality). 2 Two-way verifications (Client and server verify each other's legality).
<cs_id>	Hex integer type. Start with 0x. 0xFFFF supports all cipher suites. 0x0004 TLS_RSA_WITH_RC4_128_MD5 0x0005 TLS_RSA_WITH_RC4_128_SHA 0x002F TLS_RSA_WITH_AES_128_CBC_SHA 0x0035 TLS_RSA_WITH_AES_256_CBC_SHA 0x003C TLS_RSA_WITH_AES_128_CBC_SHA256 0x003D TLS_RSA_WITH_AES_256_CBC_SHA256 0xc027 TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 0xc02F TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
<negotiate_time>	Integer type. Negotiation timeout. Range: 60–300. Default value: 120. Unit: second.
<SNI_value>	Integer type. Enable/disable TLS server name indication. 0 Disable 1 Enable

2.4.2. AT+QSSLCERT Upload/Download/Delete SSL Certificate

AT+QSSLCERT Upload/Download/Delete SSL Certificate	
Test Command **AT+QSSLCERT=?**	Response **+QSSLCERT: (**list of supported **<cert_type>**s**),(**range of supported **<operation_mode>**s**)** **OK**
Read Command **AT+QSSLCERT?**	Response **[+QSSLCERT: <cert_type>,<exist_flag>]** **[…]** **OK**
Write Command	Response

Upload SSL certificate **AT+QSSLCERT=<cert_type>,<operation_mode>,<length>**	**CONNECT** //After **CONNECT** is returned, input file data. The inputted data will be written into flash automatically. When the data length reaches **<length>**, the module exits data mode. **OK** Or **ERROR**
Write Command Download SSL certificate **AT+QSSLCERT=<cert_type>,<operation_mode>**	Response **+QSSLCERT: <cert_type>,<length>** //Output file data **OK** Or **ERROR**
Write Command Delete SSL certificate **AT+QSSLCERT=<cert_type>,<operation_mode>**	Response **OK** Or **ERROR**
Maximum Response Time	Determined by the speed at which users input data
Characteristics	The command takes effect immediately. The configurations are saved automatically.

Parameter

<cert_type>	String type. Certificate type. "CA" CA certificate "User Cert" Client certificate "User Key" Client key document
<operation_mode>	Integer type. Operating mode. 0 Delete SSL certificate 1 Download SSL certificate 2 Upload SSL certificate
<length>	Integer type. Length of certificate content.
<exist_flag>	Integer type. Whether certificate exists or not. 0 Does not exist 1 Exist

2.4.3. AT+QSSLOPEN Open SSL Client

This command opens an SSL client and establishes an SSL connection. Establishing an SSL connection includes a TCP three-way handshake and an SSL handshake. Hence the timeout is the sum of TCP three-way handshake timeout (120 seconds) and **<negotiate_time>**.

AT+QSSLOPEN Open SSL Client	
Test Command **AT+QSSLOPEN=?**	Response **+QSSLOPEN: (**range of supported **<SSL_ctxID>**s**),(**range of supported **<socketID>**s**),"ip/dns",(**range of **<remote_port>**s**),(**range of supported **<local_port>**s**),(**range of supported **<access_mode>**s**)** **OK**
Write Command **AT+QSSLOPEN=<SSL_ctxID>,<socketID>,<remoteIP/domain_name>,<remote_port>,<local_port>,<access_mode>**	Response If **<access_mode>** is 2: **CONNECT** Or **ERROR** If **<access_mode>** is not 2: **OK** **+QSSLOPEN: <socketID>,<err>** Or **ERROR**
Maximum Response Time	/
Characteristics	/

Parameter

<SSL_ctxID>	Integer type. SSL context ID. Range: 0–5.
<socketID>	Integer type. Socket ID. Range: 0–11.
<remoteIP>	String type. IP address of remote server.
<domain_name>	String type. Domain name of remote server.
<remote_port>	Integer type. Port of remote server. It is valid only when **<service_type>** is "TCP" /"UDP". Range: 1–65535.
<local_port>	Integer type. Local port. Range: 1–65535.
<access_mode>	Integer type. Data access mode of serial port. 0 Buffer access mode. Socket sends and receives data with AT commands. 1 Direct push Mode. Socket sends data in AT command and receives data in URC format. 2 Transparent transmission mode. Serial port is exclusively used for sending/receiving data directly to/from the Internet.
<err>	Integer type. Result code. See ***Chapter 5*** for details.

2.4.4. AT+QSSLSEND Send Data via SSL Client

AT+QSSLSEND Send Data via SSL Client	
Test Command **AT+QSSLSEND=?**	Response **+QSSLSEND: (**range of supported **<socketID>**s**),(**range of supported **<send_len>**s**)** **OK**
Write Command **AT+QSSLSEND=<socketID>,<send_len>,<data>**	Response **+QSSLSEND: <actual_send_len>** **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<socketID>	Integer type. Socket ID. Range: 0–11.
<send_len>	Integer type. Length of data to be sent. Range: 1–1460. Unit: byte.
<data>	String type. Data to be sent.
<actual_send_len>	Integer type. Actual length of data written to socket. Unit: byte.

2.4.5. AT+QSSLRECV Read Data Received by SSL Client

AT+QSSLRECV Read Data Received by SSL Client	
Test Command **AT+QSSLRECV=?**	Response **+QSSLRECV: (**range of supported **<socketID>**s**),(**range of supported **<read_len>**s**)** **OK**
Write Command **AT+QSSLRECV=<socketID>,<read_len>**	Response **+QSSLRECV: <actual_read_len>** **<data>** **OK** Or **ERROR**

Maximum Response Time	300 ms
Characteristics	/

Parameter

<socketID>	Integer type. Socket ID. Range: 0–11.
<read_len>	Integer type. Length of data to be read. Unit type.
<actual_read_len>	Integer type. Length of actually read data. Unit: byte.
<data>	String type without double quotation marks. Actually read data.

2.4.6. AT+QSSLSTATE Query SSL Client State

AT+QSSLSTATE Query SSL Client State	
Write Command **AT+QSSLSTATE=<socketID>**	Response **+QSSLSTATE: <socketID>,"SSL CLIENT",<remote_addr>,<remote_port>,<local_port>,<state>** **OK** Or **ERROR**
Read Command **AT+QSSLSTATE?**	Response Return the state of all existing connections: **[+QSSLSTATE: <socketID>,"SSL CLIENT",<remote_addr>,<remote_port>,<local_port>,<state>]** **[…]** **OK**
Execute Command **AT+QSSLSTATE**	Response Return the state of all existing connections: **[+QSSLSTATE: <socketID>,"SSL CLIENT",<remote_addr>,<remote_port>,<local_port>,<state>]** **[…]** **OK**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<socketID>	Integer type. Socket ID. Range: 0–11.
<remote_addr>	String type. IP address of remote server.
<remote_port>	Integer type. Port of remote server.
<local_port>	Integer type. Local port.
<state>	Integer type. Socket service state 0 "Initial" Client connection is not established. 1 "Opening" Client is connecting or server is trying to listen. 2 "Connected" Client connection is established. 3 "Listening" Server is listening. 4 "Closing" Client connection is closing.

2.4.7. AT+QSSLCLOSE Close SSL Client

AT+QSSLCLOSE Close SSL Client	
Test Command **AT+QICLOSE=?**	Response **+QSSLCLOSE: (**range of supported **<socketID>**s**),(**range of supported **<close_timeout>**s**)** **OK**
Write Command **AT+QSSLCLOSE=<socketID>[,<close_timeout>]**	Response **OK** **+QSSLURC: "closed",<socketID>** Or **ERROR**
Maximum Response Time	/
Characteristics	/

Parameter

<socketID>	Integer type. Socket ID. Range: 0–11.
<close_timeout>	Integer type. Timeout for closing SSL client. Range: 1–60. Default value: 10. Unit: second.

2.5. Description of MQTT-Related AT Commands

2.5.1. AT+QMTCFG Configure Optional Parameters of MQTT Client

This command configures the optional parameters of MQTT client.

AT+QMTCFG Configure Optional Parameters of MQTT Client	
Test Command **AT+QMTCFG=?**	Response **+QMTCFG: "version",(**range of supported **<clientID>**s**),(**list of supported **<vsn>**s**)** **+QMTCFG: "datatype",(**range of supported **<clientID>**s**),(**list of supported **<data_type>**s**)** **+QMTCFG: "ssl",(**range of supported **<clientID>**s**),(**list of supported **<SSL_enable>**s**),(**range of supported **<SSL_ctxID>**s**)** **+QMTCFG: "keepalive",(**range of supported **<clientID>**s**),(**range of supported **<kalive_tm>**s**)** **+QMTCFG: "session",(**range of supported **<clientID>**s**),(**list of supported **<clean_session>**s**)** **+QMTCFG: "timeout",(**range of supported **<clientID>**s**),(**range of supported **<delivery_tm>**s**),(**range of supported **<delivery_cnt>**s**),(**list of supported **<timeout_repor**t**>**s**)** **+QMTCFG: "will",(**range of supported **<clientID>**s**),(**list of supported **<will_flag>**s**),(**range of supported **<will_QoS>** s**),(**list of supported **<will_retain>**s**),"willtopic","willmessage"** **+QMTCFG: "recv/mode",(**range of supported **<clientID>** s**),(**list of supported **<recvmode>**s**)** **OK**
Write Command Configure MQTT protocol version **AT+QMTCFG="version",<clientID>[,<vsn>]**	Response If the optional parameter is omitted, query the current configuration: **+QMTCFG: "version",<clientID>,<vsn>** **OK** If the optional parameter is specified, configure the MQTT protocol version: **OK** Or **ERROR**

Write Command Configure type of data to be sent **AT+QMTCFG="datatype",<clientID>[,<data_type>]**	Response If the optional parameter is omitted, query the current configuration: **+QMTCFG: "datatype",<clientID>,<data_type>** **OK** If the optional parameter is specified, configure the type of the data to be sent: **OK** Or **ERROR**
Write Command Enable or disable SSL connection **AT+QMTCFG="ssl",<clientID>[,<SSL_enable>[,<SSL_ctxID>]]**	Response If the optional parameters are omitted, query the current configuration: **+QMTCFG: "ssl",<clientID>,<SSL_enable>[,<SSL_ctxID>]** **OK** If any of the optional parameter is specified, enable or disable SSL connection: **OK** Or **ERROR**
Write Command Configure keep-alive time of MQTT protocol **AT+QMTCFG="keepalive",<clientID>[,<kalive_tm>]**	Response If the optional parameter is omitted, query the current configuration: **+QMTCFG: "keepalive",<clientID>,<kalive_tm>** **OK** If the optional parameter is specified, configure keepalive time of MQTT protocol: **OK** Or **ERROR**
Write Command Configure session type of MQTT protocol **AT+QMTCFG="session",<clientID>[,<clean_session>]**	Response If the optional parameter is omitted, query the current configuration: **+QMTCFG: "session",<clientID>,<clean_session>** **OK**

	If the optional parameter is specified, configure session type of MQTT protocol: **OK** Or **ERROR**
Write Command Configure response waiting timeout **AT+QMTCFG="timeout",<clientID>[,<delivery_tm>,<delivery_cnt>,<timeout_report>]**	Response If the optional parameters are omitted, query the current configuration: **+QMTCFG: "timeout",<clientID>,<delivery_tm>,<delivery_cnt>,<timeout_report>** **OK** If the optional parameters are specified, configure response waiting timeout. **OK** Or **ERROR**
Write Command Configure Will information of MQTT protocol **AT+QMTCFG="will",<clientID>[,<will_flag>,<will_QoS>,<will_retain>,<will_topic>,<will_message>]**	Response If the optional parameters are omitted, query the current configuration: **+QMTCFG: "will",<clientID>,<will_flag>,<willQoS>,<will_retain>,<will_topic>,<will_message>** **OK** If the optional parameters are specified, configure Will information of MQTT protocol. **OK** Or **ERROR**
Write Command Configure how to read messages for MQTT client **AT+QMTCFG="recv/mode",<clientID>[,<recvmode>]**	Response If the optional parameter is omitted, query the current configuration: **+QMTCFG: "recv/mode",<clientID>,<recvmode>** **OK** If the optional parameter is specified, configure how to read messages for MQTT client: **OK** Or **ERROR**
Maximum Response Time	300 ms

Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<clientID>	Integer type. MQTT client number. Range: 0–5.
<vsn>	Integer type. MQTT protocol version. 3 MQTT protocol V3 4 MQTT protocol V4
<data_type>	Integer type. Type of data to be sent. 0 Non-hex string 1 Hex string
<SSL_enable>	Integer type. Enable or disable SSL. 0 Disable. Use Raw TCP connection for MQTT 1 Enable. Use SSL TCP secure connection for MQTT
<SSL_ctxID>	Integer type. SSL context ID. Range: 0–5.
<kalive_tm>	Integer type. Keepalive time. Maximum idle time allowed for no data interaction between the MQTT client and server. PingReq and PingResp messages are used to keep the connection between the MQTT client and the server alive. Range: 1–3600. Default value: 120. Unit: second.
<clean_session>	Integer type. Value of the field corresponding to session type in MQTT CONNECT messages. 0 Connection is considered persistent. After the client disconnects, any subscribed topics and information with QoS set to 1 or 2 are saved until the client reconnects to the server. 1 After the client disconnects, all subscribed topics will be removed.
<delivery_tm>	Integer type. Maximum time that MQTT client waits for a response from the server after sending an MQTT message. Range: 1–60. Default value: 5. Unit: second.
<delivery_cnt>	Integer type. Maximum retransmission counts of MQTT message. Range: 1–10. Default value: 3.
<timeout_report>	Integer type. Whether to report a URC when MQTT message is retransmitted. 0 Do not report 1 Report (See **AT+QMTSUB**, **AT+QMTPUB** and **AT+QMTUNS** for details.)
<will_flag>	Integer type. Value of the field corresponding to the **<will_flag>** in MQTT CONNECT messages. 0 Ignore the configurations of **<will_QoS>**,**<will_retain>**,**<will_topic>** and **<will_message>**. 1 Send the configurations of **<will_QoS>**,**<will_retain>**,**<will_topic>** and **<will_message>** to server in MQTT CONNECT message.
<will_QoS>	Integer type. Quality of service corresponding to **<will_message>**. 0 At most once 1 At least once 2 Exactly once

<will_retain>	Integer type. Whether the server permanently saves the published **<will_message>** after the MQTT client is disconnected unexpectedly. 0 Do not save 1 Save
<will_topic>	String type. Will topic name.
<will_message>	String type. Message published to the Will topic after the client is disconnected unexpectedly.
<recvmode>	Integer type. Message receiving mode published by the server. 0 Direct push mode 1 Buffer mode (Reading with **AT+QMTRECV**)

NOTE

Before connecting a client to MQTT server, you need to configure the optional parameters of MQTT client.

2.5.2. AT+QMTOPEN Open a Session Between MQTT Client and Server

This command opens a session between MQTT client and server.

AT+QIOPEN Open a Session Between MQTT Client and Server	
Test Command **AT+QMTOPEN=?**	Response **+QMTOPEN: (**range of supported **<clientID>**s**),"hostname",(**range of supported **<port>**s**)** **OK**
Read Command **AT+QMTOPEN?**	Response Returns server information of all existing connections: **[+QMTOPEN: <clientID>,<hostname>,<port>]** **[…]** **OK**
Write Command **AT+QMTOPEN=<clientID>,<hostname>,<port>**	Response **OK** **+QMTOPEN: <clientID>,<result>** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<clientID>	Integer type. MQTT client number. Range: 0–5.
<hostname>	String type. MQTT server address.
<port>	Integer type. MQTT server port. Range: 1–65535.
<result>	Integer type. Command execution results. -1 Failed to connect socket 0 MQTT session was opened successfully 1 Wrong parameter 2 **<clientID>** is occupied 3 Failed to activate PDP 4 Failed to parse domain name 5 Socket connection was closed abnormally.

2.5.3. AT+QMTCLOSE Close a Session Between MQTT Client and Server

This command closes a session between MQTT client and server.

AT+QMTCLOSE Close a Session Between MQTT Client and Server	
Test Command **AT+QMTOPEN=?**	Response **+QMTOPEN: (**range of supported **<clientID>**s**)** **OK**
Write Command **AT+QMTOPEN=<clientID>**	Response **OK** **+QMTOPEN: <clientID>,<result>** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<clientID>	Integer type. MQTT client number. Range: 0–5.
<result>	Integer type. Command execution results. -1 Failed execution 0 Successful execution

2.5.4. AT+QMTCONN Connect a Client to MQTT Server

This command connects a client to MQTT server.

AT+QMTCONN Connect a Client to MQTT Server	
Test Command **AT+QMTCONN=?**	Response **+QMTCONN: (**range of supported **<clientID>**s**),"client_identity","username","password"** **OK**
Read Command **AT+QMTCONN?**	Response Return the state of all existing connections: **[+QMTCONN: <clientID>,<state>]** **[…]** **OK**
Write Command **AT+QMTCONN=<clientID>,<client_identity>,<username>,<password>**	Response **OK** **+QMTCONN: <clientID>,<result>[,<response_code>]** Or **ERROR**
Maximum Response Time	300 ms
Characteristic	/

Parameter

<clientID>	Integer type. MQTT client number. Range: 0–5.
<client_identity>	String type. MQTT client identity.
<username>	String type. Client username.
<password>	String type. Password corresponding to client username.
<state>	Integer type. MQTT connection state. 1 MQTT is initialized 2 MQTT is connecting 3 MQTT is connected 4 MQTT is disconnecting
<result>	Integer type. Command execution result. 0 CONNECT message was sent successfully and CONNECTACK message was received. 1 CONNECT message was sent successfully but CONNECTACK message was not received within the specified response time 2 Failed to send CONNECT message

<response_code>	Integer type. Response code in CONNECTACK message. 0 Connection Accepted 1 Connection Rejected: Unacceptable Protocol Version 2 Connection Rejected: Identifier Rejected 3 Connection Rejected: Server Unavailable 4 Connection Rejected: Wrong Username or Password 5 Connection Rejected: Unauthorized

2.5.5. AT+QMTDISC Disconnect a Client from MQTT Server

This command disconnects a client from MQTT server.

AT+QMTDISC Disconnect a Client from MQTT Server	
Test Command **AT+QMTDISC=?**	Response **+QMTDISC: (**range of supported **<clientID>**s**)** **OK**
Write Command **AT+QMTDISC=<clientID>**	Response **OK** **+QMTDISC: <clientID>,<result>** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<clientID>	Integer type. MQTT client number. Range: 0–5.
<result>	Integer type. Command execution result. -1 Failed Execution. 0 Successful execution.

2.5.6. AT+QMTSUB Subscribe to Topics

This command subscribes to topic(s) published by MQTT server.

AT+QMTSUB Subscribe to Topics	
Test Command **AT+QMTSUB=?**	Response **+QMTSUB: (**range of supported **<clientID>**s**),(**range of supported **<msgID>**s**),list of["topic",<qos>]**

	OK
Write Command **AT+QMTSUB=<clientID>,<msgID>,<topic1>,<qos1>[,<topic2>,<qos2>[…]]**	Response **OK** **+QMTSUB: <clientID>,<msgID>,<result>[,<value>]** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<clientID>	Integer type. MQTT client number. Range: 0–5.
<msgID>	Integer type. SUBSCRIBE message identifier. Range: 1–65535.
<qos>	Integer type. Quality of service for **<topic>**. 0 At most once 1 At least once 2 Exactly once
<topic>	String type. Topic to be subscribed to.
<result>	Integer type. Command execution result. 0 SUBSCRIBE message was sent successfully and SUBACK message was received. 1 SUBSCRIBE message was sent successfully but SUBACK message was not received within the specified response time. Retransmission was executed. 2 Failed to send SUBSCRIBE message.
<value>	Integer type. When **<result>** is 0, it is a vector of granted QoS levels of SUBACK message. When **<result>** is 1, it indicates the number of SUBSCRIBE message retransmission. When **<result>** is 2, it is not presented.

NOTE

The command currently supports subscribing up to 5 topics at a time.

2.5.7. AT+QMTUNS Unsubscribe from Topics

This command unsubscribes from topic(s) published by MQTT server. The client sends an UNSUBSCRIBE message to the server to unsubscribe from named topics.

AT+QMTUNS Unsubscribe from Topics	
Test Command **AT+QMTUNS=?**	Response **+QMTUNS: (**range of supported **<clientID>**s**),(**range of supported **<msgID>**s**),list of["topic"]** **OK**
Write Command **AT+QMTUNS=<clientID>,<msgID>,<topic1>[,<topic2>[…]]**	Response **OK** **+QMTUNS: <clientID>,<msgID>,<result>** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<clientID>	Integer type. MQTT client identifier. Range: 0–5.
<msgID>	Integer type. UNSUBSCRIBE message identifier. Range: 1–65535.
<topic>	String type. Topic to unsubscribe from.
<result>	Integer type. Command execution result. 0 UNSUBSCRIBE message was sent successfully and UNSUBACK message was received. 1 UNSUBSCRIBE message was sent successfully but UNSUBACK message was not received within the specified response time. Retransmission was executed. 2 Failed to send UNSUBSCRIBE message.

NOTE

The command currently supports unsubscribing up to 5 topics at a time.

2.5.8. AT+QMTPUB Publish Message via MQTT Server

This command publishes messages via MQTT server.

AT+QMTPUB Publish Messages via MQTT Server	
Test Command **AT+QMTPUB=?**	Response **+QMTPUB: (**range of supported **<clientID>**s**),** **(**range of supported **<msgID>**s**),(**range of supported **<QoS>**s**),(**list of supported **<retain>**s**),"topic",(**range of supported **<payload_length>**s**),"payload"** **OK**
Write Command **AT+QMTPUB=<clientID>,<msgID>,<QoS>,<retain>,<topic>,<payload_length>,<payload>**	Response **OK** **+QMTPUB: <clientID>,<result>[,<value>]** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<clientID>	Integer type. MQTT client identifier. Range: 0–5.
<msgID>	Integer type. PUBLISH message identifier. Range: 0–65535. When QoS is 0, **<msgID>** must be 0. When QoS is greater than 0, **<msgID>** must be greater than 0.
<QoS>	Integer type. Quality of service for publishing messages. 0 At most once 1 At least once 2 Exactly once
<retain>	Integer type. After MQTT client is unexpectedly disconnected, whether the published message is to be saved on the server forever or not. 0 Eliminate without saving permanently 1 Save permanently
<topic>	String type. Topic.
<payload_len>	Integer type. Length of message to be published. Range: 1–1500. Unit: byte.
<payload>	Hexadecimal string type. Message to be published.
<result>	Integer type. Command execution result. 0 PUBLISH message was sent successfully and ACK message was received. 1 PUBLISH message was sent successfully but ACK message was not received within delivery time. Retransmission was executed.

	2 Failed to send PUBLISH message
<value>	Integer type. When **<result>** is 1, it indicates the number of PUBLISH message retransmission. When **<result>** is 0 or 2, it is not presented.

2.5.9. AT+QMTRECV Read Messages Published by MQTT Server

This command reads messages published by MQTT Server.

AT+QMTRECV Read Messages Published by MQTT Server	
Test Command **AT+QMTRECV=?**	Response **OK**
Read Command **AT+QMTRECV?**	Response **[+QMTRECV: <clientID>,<store_stauts4>,<store_stauts3>,<store_stauts2>,<store_stauts1>,<store_stauts0>]** **[…]** **OK**
Write Command **AT+QMTRECV=<clientID>[,<storeID>]**	Response If the optional parameter is omitted, read all buffered messages of the specified client: **[+QMTRECV: <clientID>,<msgID>,<topic>,<payload_len>],<payload>]** **[…]** **OK** If the optional parameter is specified, read the messages specified by **<storeID>** of the specified client: **+QMTRECV: <clientID>,<msgID>,<topic>,<payload_len>,<payload>** **OK**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<clientID>	Integer type. MQTT client identifier. Range: 0–5.
<storeID>	Integer type. ID of messages stored in buffer. Range: 0–4.
<store_status4>	Integer type. Indicates whether a message stored in buffer corresponds to storeID4. 0 No message in buffer

	1 One or more messages are stored in buffer
<store_status3>	Integer type. Indicates whether a message stored in buffer corresponds to storeID3. 0 No message in buffer 1 One or more messages are stored in the buffer
<store_status2>	Integer type. Indicates whether a message stored in buffer corresponds to storeID2. 0 No message in buffer 1 One or more messages are stored in buffer
<store_status1>	Integer type. Indicates whether a message stored in buffer corresponds to storeID1. 0 No message in buffer 1 One or more messages are stored in buffer
<store_status0>	Integer type. Indicates whether a message stored in buffer corresponds to storeID0. 0 No message in buffer 1 One or more messages stored in buffer
<msgID>	Integer type. PUBLISH message identifier. Range: 0–65535. When QoS is 0, **<msgID>** must be 0. When QoS is greater than 0, **<msgID>** must be greater than 0.
<topic>	String type. Topic received from MQTT server.
<payload_len>	Integer type. Length of received message.
<payload>	String type. Received message.

2.6. Description of HTTP(S)-Related AT Commands

HTTP requests include establishing an HTTP(S) session and interactions with HTTP(S) body; hence the maximum response time is determined by the time it takes to establish an HTTP(S) session and interactions of HTTP body. Maximum timeout of the TCP three-way handshake supported by FC41D module is 120 seconds. Maximum timeout of SSL handshake is determined by **<negotiate_time>** (Default value: 120 seconds). After sending the HTTP request, the response time is **<wait_response_time>** (Default value: 60 seconds).

2.6.1. AT+QHTTPCFG Configure Parameters for HTTP(S) Client

AT+QHTTPCFG Configure Parameters for HTTP(S) Client	
Test Command **AT+QHTTPCFG=?**	Response **+QHTTPCFG: "url",<url_string>** **+QHTTPCFG: "header",<hname>,<hvalue>** **+QHTTPCFG: "auth",<username>,<password>** **+QHTTPCFG: "response/output",(**list of supported **<output_mode>**s**)** **+QHTTPCFG: "response/header",(**list of supported **<save_header>**s**)**

	+QHTTPCFG: "sslctxid",(range of supported **<SSL_ctxID>**s**)** **OK**
Write Command Configure URL to be accessed **AT+QHTTPCFG="url"[,<url_string>]**	Response If the optional parameter is omitted, query current URL: **+QHTTPCFG: "url",<url_string>** **OK** If the optional parameter is specified, configure URL to be accessed: **OK** Or **ERROR**
Write Command Configure/delete customized Header **AT+QHTTPCFG="header"[,<hname>[,<hvalue>]]**	Response If the optional parameters are omitted, query the current configuration: **[+QHTTPCFG: "header",<hname>,<hvalue>]** **[…]** **OK** If **<hname>** is specified and **<hvalue>** is omitted, delete the corresponding Header: **OK** Or **ERROR** If all the optional parameters are specified, configure the customized Header: **OK** Or **ERROR**
Write Command Configure basic authentication parameters **AT+QHTTPCFG="auth"[,<username>,<password>]**	Response If the optional parameters are omitted, query the current configuration: **+QHTTPCFG: "auth",<username>,<password>** **OK** If any of the optional parameters is specified, configure the basic authentication parameters:

	OK Or **ERROR**
Write Command Configure response data output mode **AT+QHTTPCFG="response/output"[,<output_mode>]**	Response If the optional parameter is omitted, query the current configuration: **+QHTTPCFG: "response/output",<output_mode>** **OK** If the optional parameter is specified, configure the output mode of response data: **OK** Or **ERROR**
Write Command Configure whether to save response header **AT+QHTTPCFG="response/header"[,<save_header>]**	Response If the optional parameter is omitted, query the current configuration: **+QHTTPCFG: "response/header",<save_header>** **OK** If the optional parameter is specified, configure whether to save response header: **OK** Or **ERROR**
Write Command Configure SSL context ID for HTTP(S) session **AT+QHTTPCFG="sslctxid"[,<SSL_ctxID>]**	Response If the optional parameter is omitted, query the current configuration: **+QHTTPCFG: "sslctxid",<SSL_ctxID>** **OK** If the optional parameter is specified, configure SSL context ID for HTTP(S) session: **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configurations are saved automatically.

Parameter

<url_string>	String type. URL to be accessed. Empty string is returned if the parameter is not configured.
<hname>	String type. HTTP(S) Header name. Empty string is returned if the parameter is not configured.
<hvale>	String type. HTTP(S) Header value. Empty string is returned if the parameter is not configured.
<username>	String type. Username of HTTP(S) basic authentication. Empty string is returned if the parameter is not configured.
<password>	String type. User's password of HTTP(S) basic authentication. Empty string is returned if the parameter is not configured.
<output_mode>	Integer type. Response data output mode. 0 Read response data with **AT+QHTTPREAD** 1 Response data directly in URC format (See ***Chapter 3.6.5*** for details.)
<save_header>	Integer type. Whether to save response header. 0 Do not save 1 Save
<SSL_ctxID>	Integer type. SSL context ID. Range: 0–5.

2.6.2. AT+QHTTPGET Send GET Request to HTTP(S) Server

This command sends GET request to HTTP(S) server. If the GET request is sent successfully, the result of GET request is reported to MCU with **+QHTTPGET: <result>[,<status_code>[,<content_length>]]**.

AT+QHTTPGET Send GET Request to HTTP(S) Server	
Test Command **AT+QHTTPGET=?**	Response **+QHTTPGET: (**range of supported **<wait_response_time>**s**)** **OK**
Execution Command **AT+QHTTPGET[=<wait_response_time>]**	Response **OK** **+QHTTPGET: <result>[,<status_code>[,<content_length>]]** Or **ERROR**
Maximum Response Time	/
Characteristics	/

Parameter

<wait_response_time>	Integer type. Server response waiting time after the GET request is sent. Range: 60–65535. Default value: 60. Unit: second.
<result>	Integer type. Result of GET request sending. 0 Success Other value Failure
<status_code>	Integer type. HTTP(S) status code. See ***Chapter 5*** for details.
<content_length>	Integer type. Length of GET request body.

2.6.3. AT+QHTTPPOST Send POST Request to HTTP(S) Server

This command sends a POST request to HTTP(S) server. If the POST request is sent successfully, the result of POST request is reported to MCU with **+QHTTPPOST: <result>[,<status_code>[,<content_length>]]**.

AT+QHTTPPOST Send POST Request to HTTP(S) Server	
Test Command **AT+QHTTPPOST=?**	Response **+QHTTPPOST: (**range of supported **<body_length>**s**),(**range of supported **<body_wait_interval>**s**),(**range of supported **<wait_response_time>**s**)[,<name>[,<file_name>[,<content_type>]]]** **OK**
Write Command Non Form-data POST **AT+QHTTPPOST=<body_length>[,<body_wait_interval>[,<wait_response_time>]]**	Response **CONNECT** //Input body. When the length of inputted data reaches **<body_length>**, the module exits data mode. **OK** **+QHTTPPOST: <result>[,<status_code>[,<content_length>]]** Or **ERROR**
Write Command Form-data POST **AT+QHTTPPOST=<body_length>,<body_wait_interval>,<wait_response_time>,<name>[,<file_name>[,<content_type>]]**	Response **CONNECT** //Input body. When the length of inputted data reaches **<body_length>**, the module exits data mode. **OK** **+QHTTPPOST: <result>[,<status_code>[,<content_length>]]** Or

	ERROR
Maximum Response Time	/
Characteristics	/

Parameter

<body_length>	Integer type. Length of POST data. Range: 1–102400. Unit: byte.
<body_wait_interval>	Integer type. Maximum waiting time for inputting body in data mode. Range: 1–65535. Default value: 60. Unit: second.
<wait_response_time>	Integer type. Server response waiting time after the POST request is sent. Range: 60–65535. Default value: 60. Unit: second.
<name>	String type. Form-data name.
<file_name>	String type. Name of file stored on HTTP(S) server after data is uploaded.
<content_type>	String type. File content type.
<result>	Integer type. Result of POST request sending. 0 Success Other values Failure
<status_code>	Integer type. HTTP(S) status mode. See ***Chapter 5*** for details.
<content_length>	Integer type. Length of POST request body.

2.6.4. AT+QHTTPPUT Send PUT Request to HTTP(S) Server

This command sends a PUT request to HTTP(S) server. If the PUT request is sent successfully, the result of PUT request is reported to MCU with **+QHTTPPUT: <result>[,<status_code>[,<content_length>]]**.

AT+QHTTPPUT Send PUT Request to HTTP(S) Server	
Test Command **AT+QHTTPPUT=?**	Response **+QHTTPPUT: (**range of supported **<body_length>**s**),(**range of supported **<body_wait_interval>**s**),(**range of supported **<wait_response_time>**s**)** **OK**
Writer Command **AT+QHTTPPUT=<body_length>[,<body_wait_interval>[,<wait_response_time>]]**	Response **CONNECT** //Input body. When the length of inputted data reaches **<body_length>**, the module exits data mode. **OK** **+QHTTPPUT: <result>[,<status_code>[,<content_length>]]**

	Or **ERROR**
Maximum Response Time	/
Characteristics	/

Parameter

<body_length>	Integer type. Length of PUT data. Range: 1–102400. Unit: byte.
<body_wait_interval>	Integer type. Maximum waiting time for inputting body in data mode. Range: 1–65535. Default value: 60. Unit: second.
<wait_response_time>	Integer type. Server response waiting time after the PUT request is sent. Range: 60–65535. Default value: 60. Unit: second.
<result>	Integer type. Result of PUT request sending. 0 Success Other value Failure
<status_code>	Integer type. HTTP(S) status mode. See ***Chapter 5*** for details.
<content_length>	Integer type. Length of PUT request body.

2.6.5. AT+QHTTPREAD Read Response Data of HTTP(S) Request

This command reads the response data of HTTP(S) request. If the request is successful and the server responds, the output mode of response data is configured with **AT+QHTTPCFG="response/output",0**.

AT+QHTTPREAD Read Response Data of HTTP(S) Request	
Test Command **AT+QHTTPREAD=?**	Response **+QHTTPREAD: (**range of supported **<wait_response_interval>**s**)** **OK**
Write/Execution Command **AT+QHTTPREAD[=<wait_response_interval>]**	Response **CONNECT** **<data>** **OK** **+QHTTPREAD: <result>** Or **ERROR**
Maximum Response Time	/
Characteristics	/

Parameter

<wait_response_interval>	Integer type. Maximum time module serial port waits for server response. Range: 60–65535. Default value: 60. Unit: second.
<result>	Integer type. Result of response data reading. 0 Success Other values Failure
<data>	String type without double quotation marks. Read data.

3 Description of URCs

3.1. Wi-Fi-Related URCs

3.1.1. +QSTASTAT URC Indicating Station State Change

+QSTASTAT URC Indicating Station State Change	
+QSTASTAT:<event>	This URC indicates STA state changes.

Parameter

<event>	Integer type without double quotation marks. Event reported when STA state changes.	
	WLAN DISCONNECTED	Disconnected
	WLAN_CONNECTED	Connected
	GOT_IP	Got IP
	SCAN_NO_AP	Scanned no AP

3.2. BLE-Related URCs

3.2.1. +QBLESTAT URC Indicating BLE State Change

+QBLESTAT URC Indicating BLE State Change	
+QBLESTAT: <event>	This URC indicates BLE state changes.

Parameter

<event>	String type without double quotation marks. Event reported when BLE state changes.	
	NOINIT	Uninitialized event
	INIT	Initialized event
	ADVERTISING	Advertising event
	NOADVERTISING	Not advertising event

	CONNECTED	Connected event
	DISCONNECTED	Disconnected event

3.2.2. +QBLEMTU URC Indicating BLE MTU Change

+QBLEMTU URC Indicating BLE MTU Change	
+QBLEMTU: <MTU_value>	This URC indicates MTU changes of BLE.

Parameter

<MTU_value>	Integer type. Maximum transmission unit value. Range: 23–512. Unit: byte.

3.2.3. +QBLEINFO URC Indicating Handle and Characteristic UUID of Connected BLE Device

When the BLE connects other devices successfully as the central device, the URC is reported to display the handle and characteristic UUID of the connected BLE device.

+QBLEINFO URC Indicating Handle and Characteristic UUID of Connected BLE Device	
+QBLEMTU: <handle> <UUID>	The URC indicates the handle and characteristic UUID of the connected BLE device.

Parameter

<handle>	Integer type. Handle of connected BLE device. Range: 0–255.
<UUID>	Integer type. Characteristic UUID of connected BLE device. Length: 2 bytes or 16 bytes.

3.3. TCP/UDP-Related URCs

3.3.1. +QIOPEN URC Indicating TCP/UDP Socket Service Opening Result

After opening a socket service with **AT+QIOPEN**, the URC is reported to indicate the result of socket service opening.

+QIOPEN URC Indicating TCP/UDP Socket Service Opening Result	
+QIOPEN: <socketID>,<err>	The URC indicates the result of opening a TCP/UDP socket service.

Parameter

<socketID>	Integer type. Socket ID. Range: 0–11.
<err>	Integer type. Result code. See ***Chapter 5*** for details.

3.3.2. +QIURC: "recv" URC Indicating Incoming Data

The URC is reported after TCP/UDP socket service receives data.

+QIURC: "recv" URC Indicating Incoming Data	
+QIURC: "recv",<socketID>	The URC indicates incoming data in buffer access mode.
+QIURC: "recv",<socketID>,<data_len><CR><LF><data>	The URC indicates incoming data in direct push mode when **<service_type>** is not "UDP SERVICE".
+QIURC: "recv",<socketID>,<data_len>,<remote_addr>,<remote_port><CR><LF><data>	The URC indicates incoming data in direct push mode when **<service_type>** is "UDP SERVICE".

Parameter

<socketID>	Integer type. Socket ID. Range: 0–11.
<data_len>	Integer type. Data length. Range: 1–1500. Unit: byte.
<remote_addr>	String type. Source address of data.
<remote_port>	Integer type. Source port of data.
<data>	String type without double quotation marks. Received data.

3.3.3. +QIURC: "accept" URC Indicating Incoming Connection

If the acceptance mode of incoming connection is set to manual with **AT+QICFG="accept/mode"**, the URC is reported as an incoming connection is received.

+QIURC: "accept" URC Indicating Incoming Connection	
+QIURC: "accept",<socketID>	The URC indicates that an incoming connection is received if the acceptance mode of incoming connection is set to manual.

Parameter

<socketID>	Integer type. Socket ID. Range: 0–11.

3.3.4. +QIURC: "closed" URC Indicating Connection Closed

When TCP/UDP socket service is closed, the URC will be reported, and the state of socket service will be "closing".

+QIURC: "closed" URC Indicating Connection Closed	
+QIURC: "closed",<socketID>	The URC indicates that TCP/UDP socket service is closed.

Parameter

<socketID>	Integer type. Socket ID. Range: 0–11.

3.3.5. +QIURC: "incoming" URC Indicating Incoming Connection

If the acceptance mode of incoming connection is set to automatic with **AT+QICFG="accept/mode"**, the URC is reported as an incoming connection is received.

+QIURC: "incoming" URC Indicating Incoming Connection	
+QIURC: "incoming",<incoming_socketID>,<listener_socketID><remote_addr>,<remote_port>	The URC indicates that an incoming connection is received If the acceptance mode of incoming connection is set to automatic.

Parameter

<incoming_socketID>	Integer type. Socket ID of TCP server. Range: 0–11.
<listener_socketID>	Integer type. Socket ID of TCP server. Range: 0–11.
<remote_addr>	String type. Source address of incoming connection.
<remote_port>	Integer type. Source port of incoming connection.

3.3.6. +QIURC: "incoming full" URC Indicating a Full Incoming Connection

If the incoming connection reaches the limit, or no socket system resources can be allocated, then the module will report the URC as **+QIURC: "incoming full"** for the new incoming connection request.

+QIURC: "incoming full" URC Indicating a Full Incoming Connection	
+QIURC: "incoming full"	The URC indicates that the incoming connection is full.

3.3.7. NO CARRIER URC Indicating Abnormal Disconnection in Transparent Transmission Mode

NO CARRIER URC Indicating Abnormal Disconnection in Transparent Transmission Mode	
NO CARRIER	The URC indicates an abnormal disconnection in transparent transmission mode.

3.4. SSL Related URCs

3.4.1. +QSSLOPEN URC Indicating SSL Client Opening Result

After opening an SSL client with **AT+QSSLOPEN**, the URC is reported to indicate the result of SSL client opening.

+QSSLOPEN URC Indicating SSL Client Opening Result	
+QSSLOPEN: <socketID>,<err>	The URC indicates the result of SSL client opening.

Parameter

<socketID>	Integer type. Socket ID. Range: 0–11.
<err>	Integer type. Result code. See ***Chapter 5*** for details.

3.4.2. +QSSLURC: "recv" URC Indicating Incoming Data

The URC is reported when the SSL client receives data.

+QSSLURC: "recv" URC Indicating Incoming Data	
+QSSLURC: "recv",<socketID>	The URC indicates incoming data in buffer access mode.
+QSSLURC: "recv",<socketID>,<data_len><CR><LF><data>	The URC indicates incoming data in URC mode.

Parameter

<socketID>	Integer type. Socket ID. Range: 0–11.
<data_len>	Integer type. Length of received data. Range: 1–1500. Unit: byte.
<data>	String type without double quotation marks. Received data.

3.4.3. +QSSLURC: "closed" URC Indicating SSL Client Closed

When the SSL client is closed, the URC will be reported, and the state of SSL client will be "Closing".

+QSSLURC: "closed" URC Indicating SSL Client Closed	
+QSSLURC: "closed",<socketID>	The URC indicates that the SSL client is closed.

Parameter

<socketID>	Integer type. Socket ID. Range: 0–11.

3.5. MQTT Related URCs

3.5.1. +QMTRECV URC Indicating Incoming Message

The URC is reported when MQTT client receives new messages.

+QMTRECV URC Indicating Incoming Message	
+QMTRECV: <clientID>,<storeID>	The URC is reported when the client receives messages in buffer mode (**<recvmode>**=1).
+QMTRECV: <clientID>,<msgID>,<topic>,<payload_len>,<payload>	The URC is reported when the client receives messages in direct push mode (**<recvmode>**=0).

Parameter

<clientID>	Integer type. MQTT client identifier. Range: 0–5.
<storeID>	Integer type. ID of messages stored in buffer. Range: 0–4.
<msgID>	integer type. PUBLISH message identifier. Range: 0–65535.
<topic>	String type. Topic received from server.
<payload_len>	Integer type. Length of received message.
<payload>	String type. Received message.

3.5.2. +QMTSTAT URC Indicating Abnormal Disconnection from MQTT Client

The URC is reported when MQTT client is disconnected abnormally.

+QMTSTAT URC Indicating Abnormal Disconnection from MQTT Client	
+QMTSTAT: <clientID>,<stat>	The URC indicates that the MQTT client is disconnected abnormally.

Parameter

<clientID>	Integer type. MQTT client identifier. Range: 0–5.
<stat>	Integer type. Reason for abnormal disconnection of a MQTT session. 1 MQTT session is closed passively. 2 MQTT session is closed due to PingReq message timeout. 3 MQTT session is closed due to Connect message timeout. 4 MQTT session is closed due to a prompt of connection failure in ConnACK.

3.6. HTTP(S) Related URCs

3.6.1. +QHTTPGET URC Indicating Result of GET Request Sending

After sending a GET request with **AT+QHTTPGET**, the URC is reported to indicate the result of GET request sending.

+QHTTPGET URC Indicating Result of GET Request Sending	
+QHTTPGET: <result>[,<status_code>[,<content_length>]]	The URC is reported to indicate the sending result of GET request.

Parameter

<result>	Integer type. Result of GET request sending. 0 Success Other value Failure
<status_code>	Integer type. HTTP(S) status code. See ***Chapter 5*** for details.
<content_length>	Integer type. Length of GET request body.

3.6.2. +QHTTPPOST URC Indicating Result of POST Request Sending

After sending a POST request with **AT+QHTTPPOST**, the URC is reported to indicate the result of POST request sending.

+QHTTPPOST URC Indicating Result of POST Request Sending	
+QHTTPPOST: <result>[,<status_code>[,<content_length>]]	The URC is reported to indicate the result of POST request sending.

Parameter

<result>	Integer type. Result of POST request sending. 0 Success Other value Failure
<status_code>	Integer type. HTTP(S) status code. See ***Chapter 5*** for details.
<content_length>	Integer type. Length of POST request body.

3.6.3. +QHTTPPUT URC Indicating Result of PUT Request Sending

After sending a PUT request with **AT+QHTTPPUT**, the URC is reported to indicate the result of PUT request sending.

+QHTTPPUT URC Indicating Result of PUT Request Sending	
+QHTTPPUT: <result>[,<status_code>[,<content_length>]]	The URC is reported to indicate the result of PUT request sending.

Parameter

<result>	Integer type. Result of PUT request sending. 0 Success Other value Failure
<status_code>	Integer type. HTTP(S) status code. See ***Chapter 5*** for details.
<content_length>	Integer type. Length of PUT request body.

3.6.4. +QHTTPREAD URC Indicating Result of Response Data Reading

After reading the response data with **AT+QHTTPREAD**, the URC is reported to indicate the reading result.

+QHTTPREAD URC Indicating Result of Response Data Reading	
+QHTTPREAD: <result>	The URC is reported to indicate the result of the response data reading.

Parameter

<result>	Integer type. Result of response data reading. 0 Success Other value Failure

3.6.5. +QHTTPURC: "recv" URC Indicating Incoming Response Data

The URC is reported when the HTTP(s) client receives response data.

+QHTTPURC: "recv" URC Indicating Incoming Response Data	
+QHTTPURC: "recv",<length><CR><LF><data>	The URC indicates incoming response data.

Parameter

<length>	Integer type. Length of received data.
<data>	String type without double quotation marks. Response data.

4 Examples

4.1. Wi-Fi Function

```
AT+QSTAAPINFO=testssid,123456789      //Set the module to STA mode and connect to AP hotspot.
OK
AT+QSOFTAP=testap,12345678            //Enable AP mode.
OK
```

4.2. BLE Function

4.2.1. Peripheral Role

The LE device, which accepts the request to establish an active physical connection, is a peripheral device. When the connection is established, the peripheral device operates as a slave in the link layer.

4.2.1.1. Set Module to a Peripheral Device

```
AT+QBLEINIT=2                         //Set the module as a peripheral device for initializing BLE.
OK
AT+QBLENAME=QuecFC41D                 //Set BLE name.
OK
AT+QBLEADDR?                          //Query and obtain BLE device address.
+QBLEADDR:c8:47:8c:42:00:49

OK
AT+QBLEGATTSSRV=fff1                  //Establish a BLE service and set the service UUID to fff1.
OK
AT+QBLEGATTSCHAR=fff2                 //Set GATT characteristic UUID to fff2.
OK
AT+QBLEGATTSCHAR=fff3                 //Set GATT characteristic UUID to fff3.
OK
AT+QBLEADVPARAM=150,150               //Set BLE advertising parameters.
OK
```

```
AT+QBLEADVSTART                    //Start BLE advertising.
OK
```

4.2.1.2. nRF Connect

1. First, open nRF Connect application. Next, click "**SCAN**" to scan peripherals and choose "**QuecFC41D**" in the scanning results. Then click "**CONNECT**":

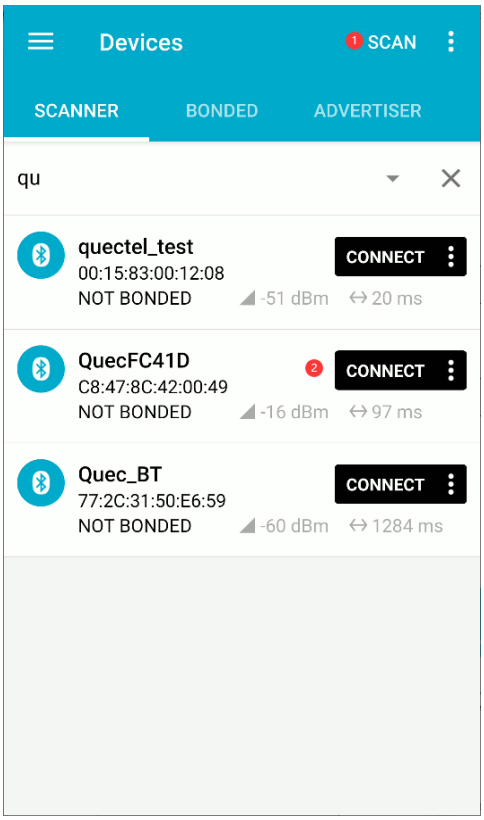

Figure 2: Scanning Result

2. After the module is connected successfully, "**CONNECTED**" is displayed in the interface, and the added UUID is displayed in "**CLIENT**":

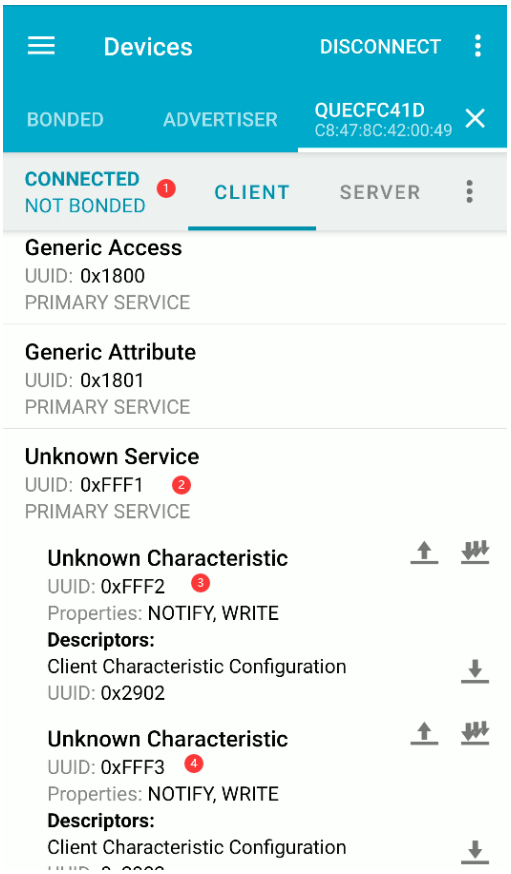

Figure 3: Connected Successfully

4.2.1.3. Send Data to Module

1. Select an editable characteristic, and click the up arrow:

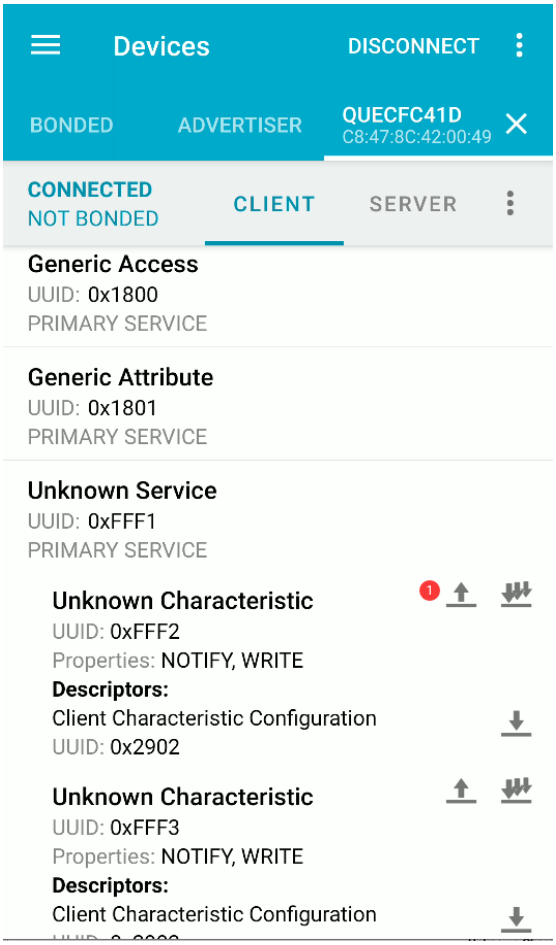

Figure 4: Edit Characteristic

2. Input the data to be sent in *TEXT* format. Then click "**SEND**":

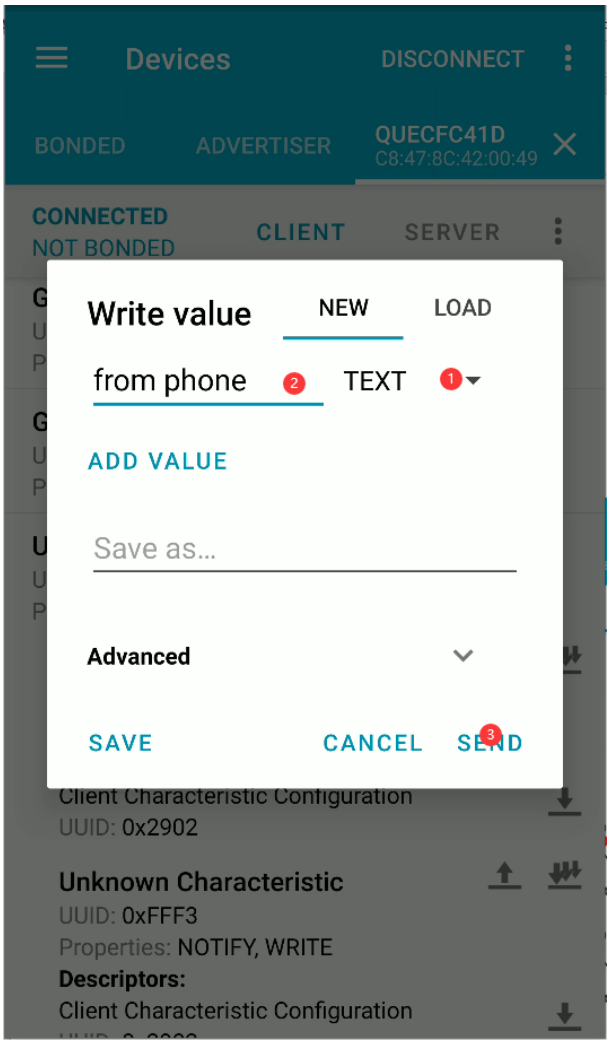

Figure 5: Send Data

3. Once the data is sent successfully, QCOM tool receives the data:

```
from phone
```

4.2.1.4. Send Data to nRF Connect

1. Enable the notification function of UUID 0xFFF3 and use QCOM tool to send data. Example:

```
AT+QBLEGATTSNTFY=fff3,from fc41d
OK
```

2. The received data is displayed in nRF Connect UUID 0xFFF3:

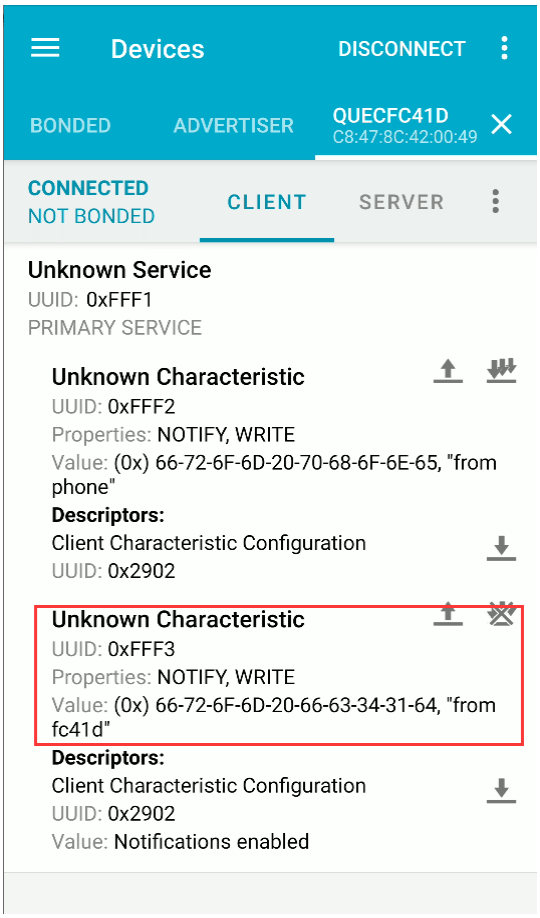

Figure 6: Received Data in 0xFFF3

4.2.2. Central Role

The LE device, which initiates the request to establish an active physical connection, is a central device. Once the connection is established the central device operates as a master in the link layer.

4.2.2.1. Set Module as Central Device

```
AT+QBLEINIT=1                       //Set the module as the central device for initializing BLE.
OK
AT+QBLESCAN=1                       //Start BLE scan.
OK

+QBLESCAN: HTV33,1,90d4c4c51a65
AT+QBLESCAN=0                       //Stop BLE scan.
OK
AT+QBLECONN=1,90d4c4c51a65          //Connect a peripheral device.
OK
+QBLESTAT: CONNECTED
+QBLEINFO: 3   2a05
+QBLEINFO: 22   2a00
+QBLEINFO: 24   2a01
+QBLEINFO: 26   2aa6
+QBLEINFO: 42   fff2
```

4.2.2.2. Configure nRF Connect

4.2.2.2.1. Configure Service

1. Start nRF Connect application. Then click "**Configure GATT server**":

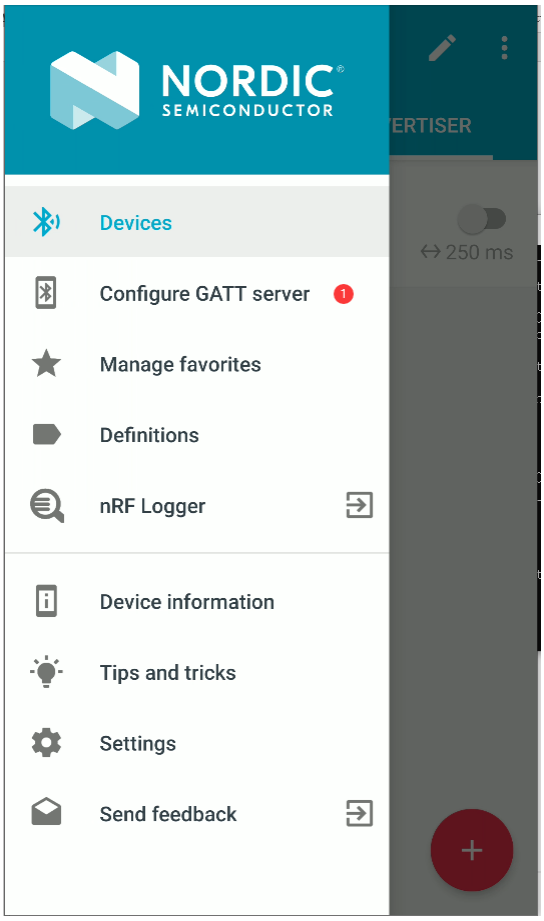

Figure 7: Configure Servce

2. Add a device configuration named "**FC14D**". Set service UUID to 0xFFF1 and characteristic UUID to 0xFFF2.

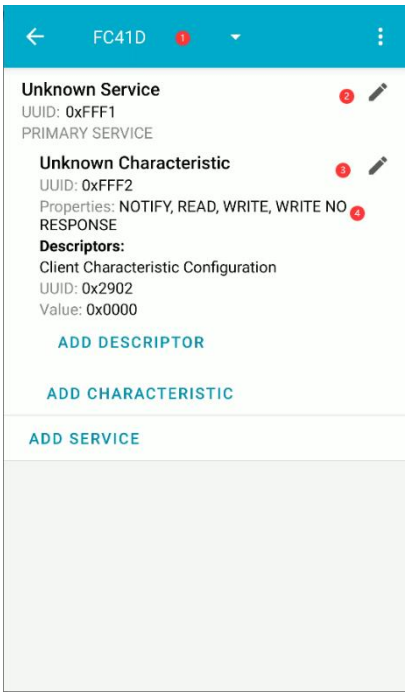

Figure 8: Configure UUID

4.2.2.2.2. Configure Advertising

1. Select "**ADVERTISER**" and click the plus sign in the lower right corner.

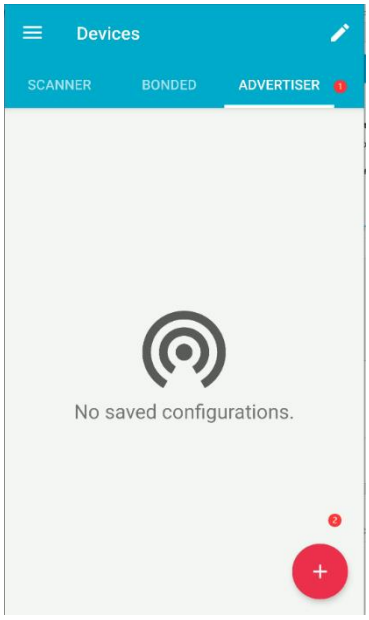

Figure 9: Add Advertiser

2. Input "**FC41D**" in "**Display name**" and tick "**Connectable**" in "**Options**". Then click "**OK**".

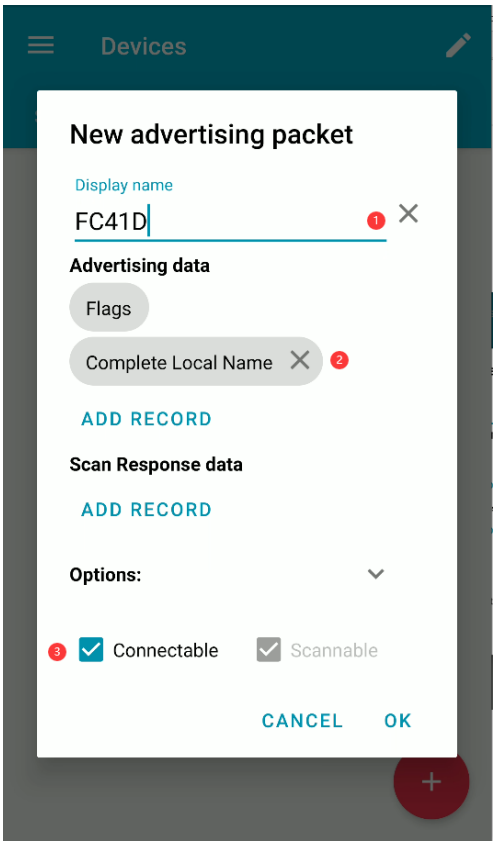

Figure 10: Configure Advertising

3. Advertising starts after the configuration is completed. You can set the advertisement duration manually.

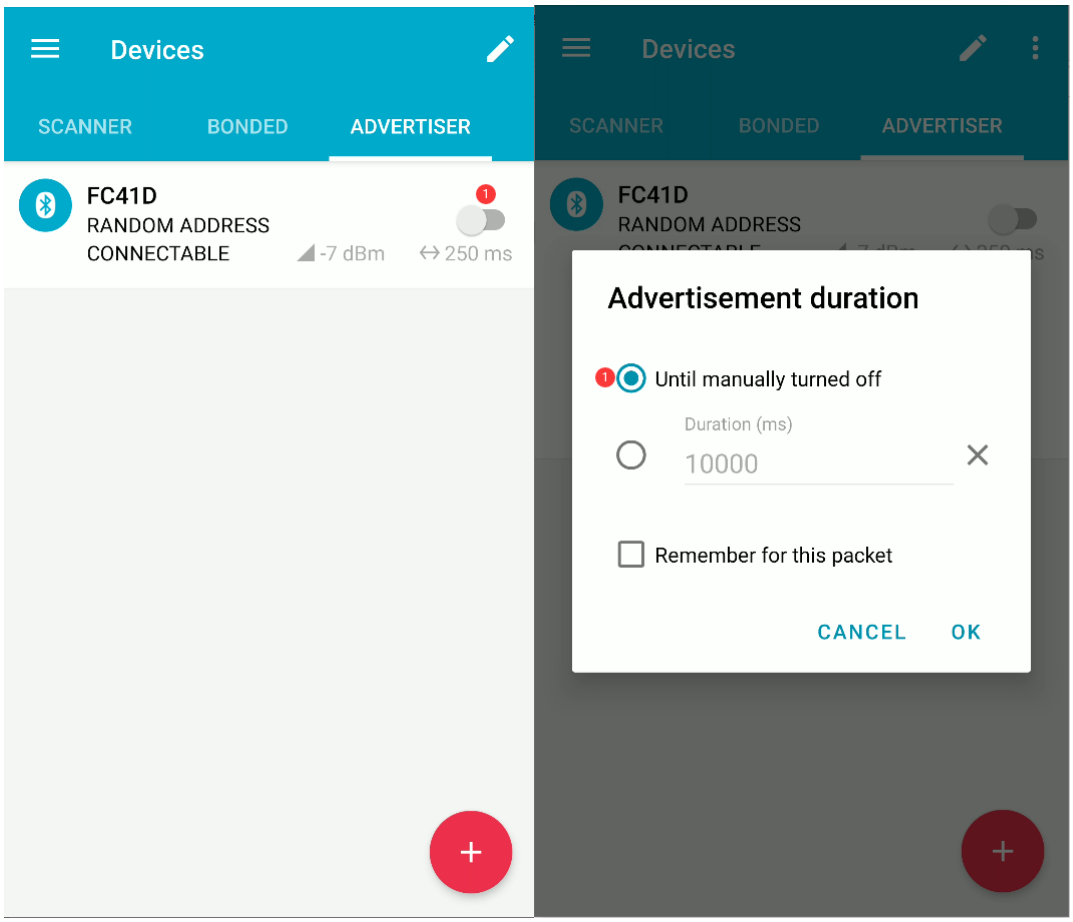

Figure 11: Set Advertisement Duration Manually

4.2.2.3. Send Data to Module

1. Select an editable characteristic, and click the up arrow:

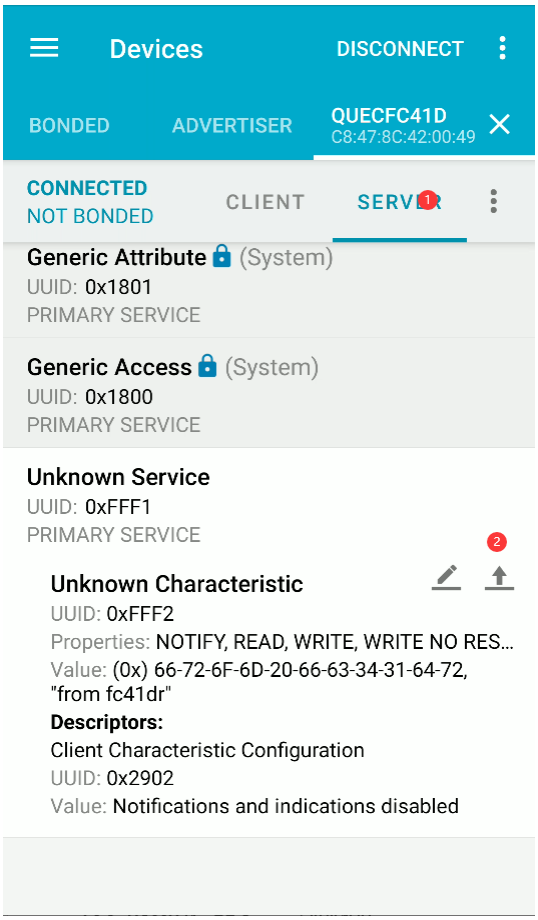

Figure 12: Edit Characteristic

2. Input the data to be sent in *TEXT* format. Then click "**SEND**":

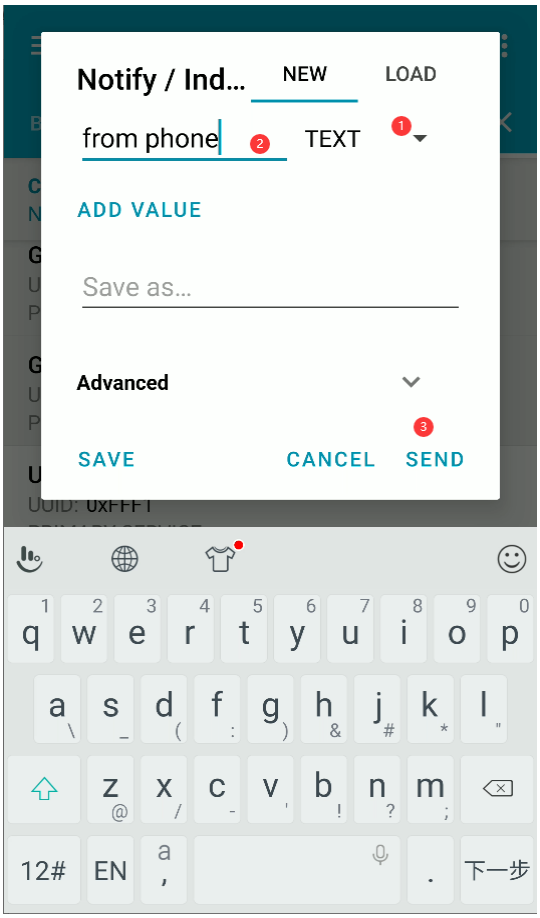

Figure 13: Send Data

3. After the data is sent successfully, QCOM tool receives the data:

```
from phone
```

4.2.2.4. Send Data to nRF Connect

1. Enable the notification function of UUID 0xFFF2 and use QCOM tool to send data. Example:

```
AT+QBLEGATTCWR=fff2,from  fc41dr
OK
```

2. The received data is displayed in nRF Connect UUID 0xFFF2:

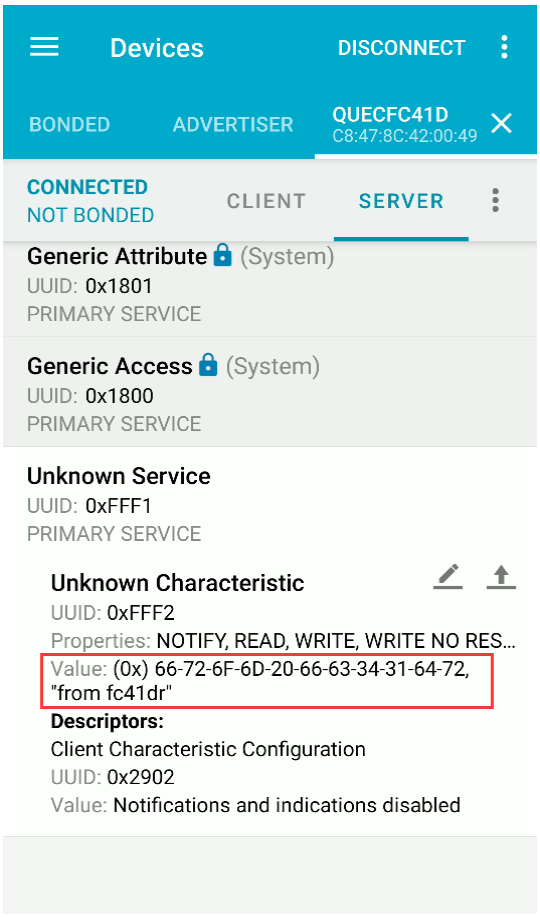

Figure 14: Received Data in 0xFFF2

4.2.3. Configure Wi-Fi via BLE

4.2.3.1. Preparation

Prepare two modules, module A and module B. Configure module A as a peripheral device and module B as the central device, and then connect module A to module B.

4.2.3.1.1. Configure Module A as a Peripheral for Configuring Wi-Fi

```
AT+QBLENAME=Quec_FC41D          //Set BLE name.
OK
AT+QBLEINIT=3                   //Set the module as a peripheral device for configuring Wi-Fi
                                  through BLE.
OK
```

4.2.3.1.2. Configure Module B as the Central Device

Open the QCOM tool, select "**COM Port**" and click "**Open Port**". Then reset FC41D.

```
AT+QBLEINIT=1                          //Set the module as the central device for initializing BLE.
OK
AT+QBLESCAN=1                          //Start BLE Scan.
OK

+QBLESCAN: Quec_FC41D,0,bcd10cf0fb80
OK
AT+QBLESCAN=0                          //Stop BLE Scan.
OK
AT+QBLECONN=0,bcd10cf0fb80             //Connect a peripheral device.
```

4.2.3.2. Configure Wi-Fi via BLE

The module B sends an AT command through **AT+QBLEGATTCWR=<UUID>,<data>** to configure Wi-Fi. **<data>** is a Wi-Fi-related AT commands. If there are multiple parameters in the configured AT commands, you need to add "****" before "**,**". For example, **AT+QBLEGATTCWR=<UUID>,AT+QSTAAPINFO=<SSID>\,<pwd>**.

```
AT+QBLECFGMTU=512                                   //Update MTU.
OK
+QBLEMTU: <512>
AT+QBLEGATTCWR=ff01,AT+QWSCAN                       //Send AT+QWSCAN via BLE.
OK
+QWSCAN:"Quectel-Customer-2.4G",WPA2_MIXED_PSK,32,f0:9b:b8:32:94:b0 ,9
+QWSCAN:"Quectel-HF-2.4G",UNKNOWN,32,f0:9b:b8:32:94:b2 ,9
+QWSCAN:"Quectel-HF",UNKNOWN,32,f0:9b:b8:32:94:b3 ,9
+QWSCAN:"ST_ShortRange",WPA2_AES_PSK,29,e0:d4:62:13:93:30 ,6
+QWSCAN:"hellowr",WPA2_AES_PSK,26,90:bd:e6:e3:66:9e ,6
+QWSCAN:"Quectel-HF",UNKNOWN,21,f0:9b:b8:33:18:b3 ,13
+QWSCAN:"Quectel-HF-2.4G",UNKNOWN,23,f0:9b:b8:33:21:f2 ,1

OK
AT+QBLEGATTCWR=ff01,AT+QSTAAPINFO=Quectel-SH\,*******  //Connect to AP hotspot via BLE.
OK
AT+QSTAAPINFO=Quectel-SH,*******OK
+QSTASTAT:WLAN_CONNECTED
+QSTASTAT:GOT_IP
```

4.3. TCP/UDP Function

4.3.1. Transparent Transmission Mode

The following example shows how to open or close a TCP/UDP client in transparent transmission mode.

```
AT+QIOPEN=0,"TCP","220.180.239.201",8252,2020,2        //Open a TCP client.
CONNECT
//Input data
+++                                                    //Exit transparent transmission mode.
OK
ATO                                                    //Enter transparent transmission mode
                                                         again.
CONNECT
//Input data
+++                                                    //Exit transparent transmission mode.
OK
AT+QICLOSE=0                                           //Close the TCP client.
OK

+QIURC: "closed",0
```

The following example shows how the TCP/UDP client in transparent transmission mode behaves if disconnected abnormally.

```
AT+QIOPEN=0,"TCP","220.180.239.201",8252,2020,2        //Open a TCP client.
CONNECT
//Input data
NO CARRIER                                             //Disconnected abnormally.
```

4.3.2. Non-transparent Transmission Mode

The following example shows how to read TCP/UDP data with AT commands.

```
AT+QIOPEN=0,"TCP","220.180.239.201",8252,2020,0        //Open a TCP client and receive data
                                                         with AT command.
OK

+QIOPEN: 0,0
AT+QISEND=0,10,"1234567890"
+QISEND: 10

OK

+QIURC: "recv",0                                       //Receive new data.
```

```
AT+QIRD=0,10                                                    //Read new data.
+QIRD:10

0123456789

OK
AT+QICLOSE=0                                                    //Close the TCP client.
OK

+QIURC: "closed",0
AT+QIOPEN=1,"UDP SERVICE","220.180.239.201",8252,2020,0         //Open a UDP client and receive
                                                                  data with AT command.
OK

+QIOPEN: 1,0
AT+QISEND=1,10,"1234567890","220.180.239.201",8252
+QISEND: 10

OK

+QIURC: "recv",1                                                //Receive new data.
AT+QIRD=1,10                                                    //Read new data.
+QIRD:10,"220.180.239.201",8252

0123456789

OK
AT+QICLOSE=1                                                    //Close the UDP service.
OK

+QIURC: "closed",1
```

The following example shows how to report new data with URC.

```
AT+QIOPEN=0,"TCP","220.180.239.201",8252,2020,1                 //Open a TCP client and receive data
                                                                  in URC format.
OK

+QIOPEN: 0,0
AT+QISEND=0,10,"1234567890"
+QISEND: 10

OK

+QIURC: "recv",0,10                                             //Receive new data.
```

```
0123456789
AT+QICLOSE=0                                                    //Close the TCP client.
OK

+QIURC: "closed",0
AT+QIOPEN=1,"UDP SERVICE","220.180.239.201",8252,2020,1 //Open a UDP client and receive
                                                                data in URC format.
OK

+QIOPEN: 1,0
AT+QISEND=1,10,"1234567890","220.180.239.201",8252
+QISEND: 10

OK

+QIURC: "recv",1,10, "220.180.239.201",8252                    //Receive new data.

0123456789
AT+QICLOSE=1                                                    //Close the UDP service.
OK

+QIURC: "closed",1
```

4.4. SSL Function

The following is an example of sending and receiving data through URC reporting during one-way verification.

```
AT+QSSLCFG="verify",1,1                                  //Set one-way verification.
OK

AT+QSSLCFG="verify",1                                    // Check whether the setting is successful.
+QSSLCFG: "verify",1,1
OK

AT+QSSLCERT="CA",2,1657                                  //Upload SSL certificate.
CONNECT

-----BEGIN CERTIFICATE-----
MIIE5TCCA02gAwIBAgIJAP5uWtPRe+IxMA0GCSqGSIb3DQEBCwUAMIGHMQswCQYD
VQQGEwJDTjELMAkGA1UECAwCQUgxCzAJBgNVBAcMAkhGMRwwGgYDVQQKDBNEZWZh
dWx0IENvbXBhbnkgTHRkMRgwFgYDVQQDDA8yMjAuMTgwLjIzOS4yMTIxJjAkBgkq
```

```
hkiG9w0BCQEWF2VkZGllLnpoYW5nQHF1ZWN0ZWwuY29tMCAXDTIxMDEyNzAzMzk0
M1oYDzIxMjEwMTAzMDMzOTQzWjCBhzELMAkGA1UEBhMCQ04xCzAJBgNVBAgMAkFl
MQswCQYDVQQHDAJlRjEcMBoGA1UECgwTRGVmYXVsdCBDb21wYW55IEx0ZDEYMBYG
A1UEAwwPMjIwLjE4MC4yMzkuMjEyMSYwJAYJKoZIhvcNAQkBFhdlZGRpZS56aGFu
Z0BxdWVjdGVsLmNvbTCCAaIwDQYJKoZIhvcNAQEBBQADggGPADCCAYoCggGBANf3
w0ep+Sv4qDjhafwc3wiaGdzwAXOsRgfGgBEGYL5MuPbpzjERo21yHae6Tx19DIUt
g6hgW5N/bvDqXLbEMChy9b8aNl62+Y6O35Tya2hoz4XXQkrPkRXvlP2eonjPMfQB
Yu4wtAfNKPa10MJF8qjFI7WWBpcQMpoQvt/SekoD6lyxgRu8ApjtmJicvvWu3BqW
W93MZqLhSFf57pKLQEhgskpl/gb+rLiUwObMTRM948J04NoYBx1jwMZLNEFs6os6
91J8B5oHlgQAsgcXjoXORxncgt2d/fZyr9NjVevc/GIkcbfVAAWohFCQmjpgLESW
iTPaRlLyeLLjTb+WrSOKosinVH+1R8ozfBnkzO+tf9lpBzdP9QCDYzNQeU4Xp0wJ
8Neo5tA4tioDEZjkbsHL+bVvssqOwfupR9b03Z2ZLfgEzs3TwYbPB7ULJQrdqe9x
QtpaFSsDeyKJ6CR8yEHz5d0lypPeRLCjZGb49Yo3FKhMdzP+ZMT+Ku001cr4GwID
AQABo1AwTjAdBgNVHQ4EFgQU4FfBsWrgpoGtvuF+3XN9kbUx0yUwHwYDVR0jBBgw
FoAU4FfBsWrgpoGtvuF+3XN9kbUx0yUwDAYDVR0TBAUwAwEB/zANBgkqhkiG9w0B
AQsFAAOCAYEAqfVX0LhhxYZ/KsC9jhWYHgmYggVcsTa7AtZsNKW79TLmz98iWdfM
wpi44rwbn1Xek5pDH/rimaJuwxcX8Q4cFHqeQJStRkcuU6CgyurVBpjGWRv5qt3W
nJc/z92x3TPzW1VJv5rXj4pzX9N4hlAHGGzFBm+VMAexS006/dksGa9uEKE/2A0+
9W/V9YbcjhejdENiRAvJB4J0QsOrNUjsH5bPEa3CxdXbKOQzGjJtS7f0BH38Fmyi
C+Cui8U0c+BwRGY3HXL7ANhCe0vdUbUGCG2L6byRvf1TlkuGpi0RxtQfEF3sTDH5
jAot50rJhbckQyLH0xkIOQ9qmU/gbt/wgoZ9AzUVIyh0RsyWo19BGz2DpsuYNBXD
4jqL4NMqsyGRq5YUTrJlli9PVUp176Ec79xSffvUbitiq9fMmxuhsRbkP4piM1TE
D5oXKPme86RvR1/foRqAdbJg5RPYdah3LdOlAE2HePVy6b0xQ5dcCHaqHmR2SVIY
m7TQs6tfvfhy
-----END CERTIFICATE-----
OK                                                          //Finished.

AT+QSSLOPEN=1,1,"220.180.239.212 ",12000                    //Set up an SSL connection.
OK
+QSSLOPEN: 1,0

AT+QSSLSTATE                                                //Query the state of all SSL connections.

+QSSLSTATE: 1,"SSL CLIENT","220.180.239.212",12000,6601,2
OK

AT+QISWTMD=1,1                                              // Switch the data access mode to direct push mode.
OK

AT+QSSLSEND=1,6,"123456"                                    //Send data.
+QSSLSEND: 6
OK

+QSSLURC: "recv",1,6                                        // The URC indicates incoming data.
```

```
123456

AT+QSSLCERT="CA",0                                  //Deleted SSL certificate.
OK

AT+QSSLCLOSE=1,1                                    //Close SSL connection.
OK
+QSSLURC: "closed",1
```

4.5. MQTT Function

```
AT+QMTCFG="version",1,4                  //Configure MQTT protocol version to V4.
OK
AT+QMTOPEN=1,"220.180.239.212",8306      //Open an MQTT session for MQTT server.
OK

+QMTOPEN: 1,0
AT+QMTCONN=1,"client1","test","test"     //Connect a client to MQTT server.
OK

+QMTCONN: 1,0,0
AT+QMTSUB=1,1,"quectel",1                //Subscribe to topic named "quectel".
OK

+QMTSUB: 1,1,0,1
//Direct push mode
AT+QMTPUB=1,1,1,0,"quectel",3,"123"      //Publish a message with the topic "quectel".
OK

+QMTRECV: 1,1,"quectel",3,"345"          //Receive a message with the topic "quectel".
//Buffer mode
AT+QMTCFG="recv/mode",0,1
OK
AT+QMTPUB=1,1,1,0,"quectel",2,"3132"     //Publish a message with the topic "quectel".
OK

+QMTPUB: 1,1,0

+QMTRECV: 0,0
AT+QMTRECV=0,0
+QMTRECV: 1,1,"quectel",2,"12"
```

```
OK
AT+QMTUNS=1,1,"quectel"                    //Unsubscribe from topic named "quectel".
OK

+QMTUNS: 1,1,0
AT+QMTDISC=1                               //Disconnect the client from MQTT server.
OK

+QMTDISC: 1,0
```

4.6. HTTP(S) Function

```
//Example of sending HTTP(S) GET request.
//Step 1: Configure URL
AT+QHTTPCFG="url","http://www.baidu.com"
OK
//Step 2: Send GET request
AT+QHTTPGET=120                  //Open Baidu web.
OK

+QHTTPGET: 0,200
//Step 3: Read the response data
AT+QHTTPREAD=60                  //Read the response data.
CONNECT
<html>
<head>
    <script>
        location.replace(location.href.replace("https://","http://"));
    </script>
</head>
<body>
    <noscript><meta http-equiv="refresh" content="0;url=http://www.baidu.com/">
.....................
OK

+QHTTPREAD: 0
```

```
//Example of sending HTTP(S) POST request.
//Step 1：  Configure URL
AT+QHTTPCFG="url","http://220.180.239.212:8252/study_log/"
OK
//Step 2: Send POST request
```

```
AT+QHTTPPOST=1024,120,120,"file","test.txt","text/plain"          //Upload Form-data POST file.
CONNECT
//Input body. When the length of inputted data reaches <body_length>, the module exits data mode.
OK

+QHTTPPOST: 0,200,1538
//Step 3: Read the response data
AT+QHTTPREAD=60                                                   //Read the response data.
CONNECT
<html>
<head>
.....................
OK

+QHTTPREAD: 0
```

```
//Example of sending HTTP(S) PUT request.
//Step 1: Configure URL
AT+QHTTPCFG="url","http://220.180.239.212:8252/uploads/test.txt"
OK
//Step 2: Send PUT request
AT+QHTTPPUT=1024,120,120                                          //Upload file in PUT mode.
CONNECT
//Input body. When the length of inputted data reaches <body_length>, the module exits data mode.
OK

+QHTTPPUT: 0,200,1538
//Step 3: Read the response data
AT+QHTTPREAD=60                                                   //Read the response data.
CONNECT
<html>
<head>
.....................
OK

+QHTTPREAD: 0
```

```
//Customized Header Example
//Customizing the parameter header: Range: bytes=x-x, to run the function of breakpoint download
AT+QHTTPCFG="header","Range","bytes=0-511"
OK
AT+QHTTPCFG="url","http://116.247.104.27:6023/1M.txt"
OK
AT+QHTTPGET=60
```

```
OK

+QHTTPGET: 0,200,512
AT+QHTTPREAD=60                                  //Read the response data.
CONNECT
//Data
OK

+QHTTPREAD: 0
```

5 Summary of Result Codes

Table 2: TCP/UDP/SSL Result Codes

Result Code	Description
0	Operation success
550	Invalid parameter
551	Unknown error
552	Memory not enough
553	Socket ID has been used
554	Socket ID not exist
555	Socket allocate failed
556	Operation not allowed
557	Operation not supported

Table 3: HTTP(S) Result Codes

Result Code	Description
0	Operation success
1	Invalid parameter
2	Unknown error
3	Memory not enough
4	Socket failure
5	Operation not supported

6	Operation not allowed
7	No network
8	Lack of SSL Cert
9	Response timeout
10	Body wait timeout

6 Appendix References

Table 4: Terms and Abbreviations

Abbreviation	Description
ACK	Acknowledgement
AP	Access Point
BLE	Bluetooth Low Energy
BSSID	Basic Service Set Identifier
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name Server
GATT	Generic Attribute Profile
MAC	Medium Access Control
MQTT	Message Queuing Telemetry Transport
MTU	Maximum Transmission Unit
HTTP	Hyper Text Transfer Protocol
ID	Mostly refers to Identifier in terms of software
IP	Internet Protocol
OTA	Over-the-Air Technology
PSK	Pre-Shared Key
QoS	Quality of Service
RF	Radio Frequency
SNI	Server Name Indication
SSID	Service Set Identifier

SSL	Service Set Identifier
TA	Terminal Adapter
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TX	Transmit
UDP	User Datagram Protocol
URC	Unsolicited Result Code
UUID	Universally Unique Identifier
Wi-Fi	Wireless Fidelity