

深圳市新一信息技术有限公司

XY-MWC320A

# 透传固件用户指南

Ver 1.0

**Part Number:XY-MWC320A**

# 目录

|                                          |    |
|------------------------------------------|----|
| 第1章 模块使用说明 .....                         | 5  |
| 1.1 硬件连接 .....                           | 5  |
| 1.1.1 硬件准备 .....                         | 5  |
| 1.1.2 XY-MWC320A 系列 .....                | 6  |
| 第2章 AT 命令示例 .....                        | 7  |
| 2.1 AT 响应消息格式控制示例 .....                  | 7  |
| 2.1.1 启用 系统消息 过滤，实现 HTTP 透传下载功能 .....    | 7  |
| 2.2 AT+SYSMFG 命令示例 .....                 | 9  |
| 2.2.1 介绍 .....                           | 9  |
| 2.2.1.1 概述 .....                         | 9  |
| 2.2.1.2 数据类型 .....                       | 10 |
| 2.2.1.3 其他重要说明 .....                     | 11 |
| 2.2.2 出厂参数配置 .....                       | 11 |
| 2.2.2.1 查询所有出厂参数 .....                   | 12 |
| 2.2.2.2 模组名称操作 .....                     | 12 |
| 2.2.2.3 UART 配置操作 .....                  | 13 |
| 2.2.2.4 最大发射功率操作 .....                   | 14 |
| 2.2.2.5 国家代码操作 .....                     | 15 |
| 2.2.2.6 系统存储模式操作 .....                   | 17 |
| 2.2.2.7 添加自定义键值对 .....                   | 18 |
| 2.2.3 PKI 配置 .....                       | 20 |
| 2.2.3.1 SSL 客户端证书 .....                  | 21 |
| 2.2.3.1.1 查询证书信息 .....                   | 21 |
| 2.2.3.1.2 配置证书 .....                     | 22 |
| 2.2.3.1.3 使用证书 .....                     | 23 |
| 2.2.3.1.4 擦除证书 .....                     | 24 |
| 2.2.3.2 SSL 服务器证书 .....                  | 24 |
| 2.2.3.2.1 查询证书信息 .....                   | 24 |
| 2.2.3.2.2 配置证书 .....                     | 25 |
| 2.2.3.2.3 使用证书 .....                     | 26 |
| 2.2.3.2.4 擦除证书 .....                     | 28 |
| 2.2.4 GATTS 配置 .....                     | 28 |
| 2.2.4.1 查询配置信息 .....                     | 28 |
| 2.2.4.1.1 读取配置项 .....                    | 28 |
| 2.2.4.2 配置服务 .....                       | 29 |
| 2.2.4.2.1 修改现有 BLE 特征的 UUID .....        | 29 |
| 2.2.4.2.2 添加新的自定义 BLE 服务 .....           | 31 |
| 2.2.5 自定义数据存储 .....                      | 36 |
| 2.2.5.1 创建和使用自定义命名空间 .....               | 36 |
| 2.3 TCP-IP AT 示例 .....                   | 39 |
| 2.3.1 XY-MWC320A 设备作为 TCP 客户端建立单连接 ..... | 39 |
| 2.3.2 XY-MWC320A 设备作为 TCP 服务器建立多连接 ..... | 40 |
| 2.3.3 远端 IP 地址和端口固定的 UDP 通信 .....        | 42 |
| 2.3.4 远端 IP 地址和端口可变的 UDP 通信 .....        | 44 |
| 2.3.5 XY-MWC320A 设备作为 SSL 客户端建立单连接 ..... | 47 |
| 2.3.6 XY-MWC320A 设备作为 SSL 服务器建立多连接 ..... | 48 |

|                                                                          |     |
|--------------------------------------------------------------------------|-----|
| 2.3.7 XY-MWC320A 设备作为 SSL 客户端建立双向认证单连接 .....                             | 51  |
| 2.3.8 XY-MWC320A 设备作为 SSL 服务器建立双向认证多连接 .....                             | 53  |
| 2.3.9 XY-MWC320A 设备作为 TCP 客户端，建立单连接，实现 Network 透传 .....                  | 56  |
| 2.3.10 XY-MWC320A 设备作为 TCP 服务器，实现 Network 透传 .....                       | 58  |
| 2.3.11 XY-MWC320A 设备作为 softAP 在 UDP 传输中实现 Network 透传 .....               | 60  |
| 2.3.12 XY-MWC320A 设备获取被动接收模式下的套接字数据 .....                                | 62  |
| 2.3.13 XY-MWC320A 设备开启 mDNS 服务，PC 通过域名连接到设备的 TCP 服务器 .....               | 63  |
| 2.4 Bluetooth LE AT 示例 .....                                             | 65  |
| 2.4.1 简介 .....                                                           | 66  |
| 2.4.1.1 Bluetooth LE 协议架构 .....                                          | 66  |
| 2.4.1.2 Bluetooth LE 角色划分 .....                                          | 66  |
| 2.4.2 Bluetooth LE 客户端读写服务特征值 .....                                      | 67  |
| 2.4.3 Bluetooth LE 服务端读写服务特征值 .....                                      | 72  |
| 2.4.4 Bluetooth LE 连接加密 .....                                            | 79  |
| 2.4.5 两个 XY-MWC320A 开发板之间建立 SPP 连接，以及在 UART-Bluetooth LE 透传模式下传输数据 ..... | 84  |
| 2.4.6 XY-MWC320A 与手机建立 SPP 连接，以及在 UART-Bluetooth LE 透传模式下传输数据 .....      | 89  |
| 2.4.7 XY-MWC320A 和手机之间建立 Bluetooth LE 连接并配对 .....                        | 92  |
| 2.5 MQTT AT 示例 .....                                                     | 95  |
| 2.5.1 基于 TCP 的 MQTT 连接（需要本地创建 MQTT 代理）（适用于数据量少） .....                    | 95  |
| 2.5.2 基于 TCP 的 MQTT 连接（需要本地创建 MQTT 代理）（适用于数据量多） .....                    | 97  |
| 2.5.3 基于 TLS 的 MQTT 连接（需要本地创建 MQTT 代理） .....                             | 99  |
| 2.5.4 基于 WSS 的 MQTT 连接 .....                                             | 101 |
| 2.6 MQTT AT 连接云示例 .....                                                  | 104 |
| 2.6.1 从 AWS IoT 获取证书以及 endpoint .....                                    | 104 |
| 2.6.2 使用 MQTT AT 命令基于双向认证连接 AWS IoT .....                                | 104 |
| 2.6.2.1 配置证书 .....                                                       | 104 |
| 2.6.2.2 使用 AT 命令连接到 AWS IoT .....                                        | 105 |
| 2.6.2.3 示例日志 .....                                                       | 107 |
| 2.7 HTTP AT 示例 .....                                                     | 108 |
| 2.7.1 HTTP 客户端 HEAD 请求方法 .....                                           | 108 |
| 2.7.2 HTTP 客户端 GET 请求方法 .....                                            | 109 |
| 2.7.3 HTTP 客户端 POST 请求方法（适用于 POST 少量数据） .....                            | 110 |
| 2.7.4 HTTP 客户端 POST 请求方法（推荐方式） .....                                     | 112 |
| 2.7.5 HTTP 客户端 PUT 请求方法（适用于无数据情况） .....                                  | 115 |
| 2.7.6 HTTP 客户端 PUT 请求方法（推荐方式） .....                                      | 116 |
| 2.7.7 HTTP 客户端 DELETE 请求方法 .....                                         | 119 |
| 2.8 Sleep AT 示例 .....                                                    | 120 |
| 2.8.1 简介 .....                                                           | 121 |
| 2.8.1.1 测量方法 .....                                                       | 121 |
| 2.8.2 在 Wi-Fi 模式下设置为 Modem-sleep 模式 .....                                | 122 |
| 2.8.3 在 Wi-Fi 模式下设置为 Light-sleep 模式 .....                                | 123 |
| 2.8.4 在蓝牙广播态下设置为 Light-sleep 模式 .....                                    | 123 |
| 2.8.5 在蓝牙连接态下设置为 Light-sleep 模式 .....                                    | 124 |
| 2.8.6 设置为 Deep-sleep 模式 .....                                            | 126 |
| 2.9 AT 配网示例 .....                                                        | 126 |
| 2.9.1 BluFi 配网 .....                                                     | 127 |

2.9.2 SmartConfig 配网 ..... 129

    2.9.2.1 ESP-TOUCH v2 配网示例 ..... 129

    2.9.2.2 AirKiss 配网示例 ..... 130

2.9.3 SoftAP 配网 ..... 131

2.9.4 WPS 配网 ..... 131

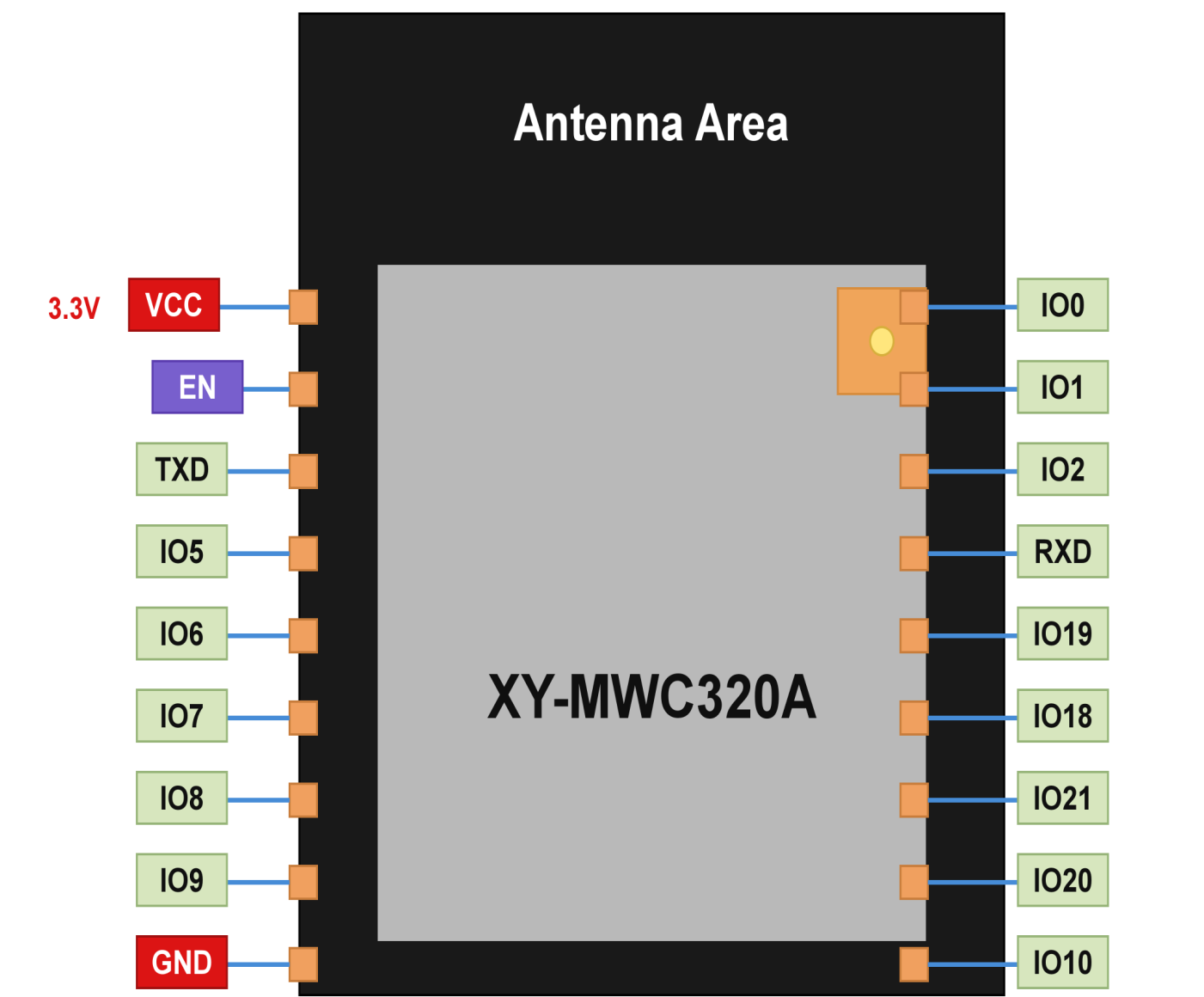
# 第1章 模块使用说明

## 1.1 硬件连接

本文档主要介绍下载和烧录 AT 固件、发送 AT 命令和接收 AT 响应所需要的硬件以及硬件之间该如何连接。

### 1.1.1 硬件准备

| 硬件                                      | 功能                    |
|-----------------------------------------|-----------------------|
| XY-MWC320A 开发板                          | 从机                    |
| USB 转 UART 串口模块                         | 转换 USB 信号和 TTL 信号     |
| 杜邦线（连接 USB 转 UART 串口模块和 XY-MWC320A 开发板） | 发送 AT 命令、接收 AT 响应数据连接 |



XY-MWC320A 模组pin 脚示意图

1.1.2 XY-MWC320A 系列

XY-MWC320A AT 采用两个 UART 接口：UART0 用于下载固件和输出日志，UART1 用于发送 AT 命令和接收 AT 响应。默认情况下，UART0 和 UART1 均使用 115200 波特率进行通信。

| 功能          | XY-MWC320A 开发板/模组管脚           | 其它设备管脚                |
|-------------|-------------------------------|-----------------------|
| 下载固件/输出日志 1 | UART0 GPIO20 (RX) GPIO21 (TX) | PC TX RX              |
| AT 命令/响应 2  | UART1 GPIO3 (RX) GPIO4 (TX)   | USB 转 UART 串口模块 TX RX |

## 第2章 AT 命令示例

### 2.1 AT 响应消息格式控制示例

#### 2.1.1 启用 系统消息 过滤，实现 HTTP 透传下载功能

本例以下载一个 PNG 格式的图片文件为例，图片链接为 <https://www.newbitinfo.com/dl/hardware.png>。

##### 1. 恢复出厂设置。

```
AT
1 AT+RESTORE
```

```
Plain Text
1 OK
```

##### 2. 设置 Wi-Fi 模式为 station。

```
AT
1 AT+CWMODE=1
```

```
Plain Text
1 OK
```

##### 3. 连接路由器。

```
AT
1 AT+CWJAP="XY-Router","1234567890"
```

```
Plain Text
1 WIFI CONNECTED
2 WIFI GOT IP
3 OK
```

**说明：**实际输入的 SSID 和密码可能与上述命令中的不同，请使用所用路由器的 SSID 和密码。

##### 4. 设置系统消息过滤器一：过滤每条 HTTP 数据的头部和尾部。

```
AT
1 AT+SYSMSGFILTERCFG=1,18,3
```

```
Plain Text
1 OK
2 >
```

此时输入 `^+HTTPCGET:[0-9]*`，（共 18 字节）和 `\r\n$`（共 3 字节，其中 `\r\n` 对应 ASCII 码中的换行和回车，即：0D 0A）。响应：

Plain Text

1 OK

**说明：** `^+HTTPCGET:[0-9]*`，为头部正则表达式，表示匹配以 `+HTTPCGET:` 开头，紧跟着一串数字，最后为逗号的字符串。`\r\n$` 为尾部正则表达式，表示匹配以 `\r\n` 结尾的字符串。

## 5. 设置系统消息过滤器二：过滤图片下载完成时的 OK 系统消息。

AT

1 AT+SYMSGFILTERCFG=1,0,7

Plain Text

1 OK

2 >

此时输入 `\r\nOK\r\n$`（共 7 字节，其中 `\r\n` 对应 ASCII 码中的换行和回车，即：0D 0A）。响应：

Plain Text

1 OK

**说明：** `\r\nOK\r\n$` 为尾部正则表达式，表示匹配以 `\r\nOK\r\n` 结尾的字符串。

## 6. 启用系统消息过滤

AT

1 AT+SYMSGFILTER=1

Plain Text

1 OK

**说明：** 只有启用系统消息过滤后，上面设置的过滤器才会生效。

## 7. 关闭回显

Plain Text

1 ATE0

Plain Text

1 OK

## 8. 下载图片

下载图片，设置发送和接收缓存大小为 2048 字节，网络超时为 5000 毫秒（注意：网络超时不是命令超时，此处的 5 秒网络超时指有连续 5 秒未接收到服务器端的数据，则关闭此网络连接。在较差

的网络环境中，如果每秒都能接收一点服务器端的数据，则不会关闭此网络连接，这可能导致命令超时很长)。

AT

```
1 AT+HTTPGET="https://www.newbitinfo.com/dl/hardware.png",2048,2048,5000
```

// 此处，MCU 将透传接收到整个 <https://www.newbitinfo.com/dl/hardware.png> 图片资源。

**说明：**如果图片下载失败，AT 仍然会发送 `\r\nERROR\r\n`（共 9 字节）系统消息给 MCU。

## 9. 清除过滤器

AT

```
1 AT+SYMSGFILTERCFG=0
```

Plain Text

```
1 OK
```

## 10. 禁用系统消息过滤

AT

```
1 AT+SYMSGFILTER=0
```

Plain Text

```
1 OK
```

## 11. 开启回显

Plain Text

```
1 ATE1
```

Plain Text

```
1 OK
```

## 2.2 AT+SYSMFG 命令示例

本文档提供使用 AT+SYSMFG 命令操作 manufacturing nvs 用户分区的详细示例。

### 2.2.1 介绍

#### 2.2.1.1 概述

manufacturing NVS 分区包含多个命名空间，每个命名空间包含若干键值对，用于存储不同类型的配置信息。按用途，可分为以下几类，每一类的命名空间、键名以及详细操作示例请参考对应子章节：

- 出厂参数配置：factory\_param 命名空间，用于存储模组信息、Wi-Fi 配置和 UART 配置等。

- PKI 配置：用于存储 SSL/MQTT/HTTPS/WebSocket/Wi-Fi Enterprise 客户端或服务器所使用的证书和私钥。
- GATTS 配置：ble\_data 命名空间，用于存储低功耗蓝牙 GATTS 服务配置。
- 自定义数据存储：用户自定义命名空间，用于存储应用特定的持久化数据。

查询当前固件实际支持的所有命名空间：

AT

```
1 AT+SYSMFG?
```

Plain Text

```
1 +SYSMFG:"factory_param"
2 +SYSMFG:"client_cert"
3 +SYSMFG:"client_key"
4 +SYSMFG:"client_ca"
5 +SYSMFG:"server_cert"
6 +SYSMFG:"server_key"
7 +SYSMFG:"server_ca"
```

...

Plain Text

```
1 +SYSMFG:"mqtt_cert"
2 +SYSMFG:"mqtt_key"
3 +SYSMFG:"mqtt_ca"
4 OK
```

## 备注

- 不同芯片和固件版本支持的命名空间可能不同。
- 本文档中所有示例的命令响应内容仅供参考，实际输出可能因芯片型号、固件版本和配置而异。

### 2.2.1.2 数据类型

AT

```
1 AT+SYSMFG 命令中的 <type> 参数表示数据类型：
```

- 1: u8 (8 位无符号整数，范围：0-255，例如：UART 端口号、信道号)
- 2: i8 (8 位有符号整数，范围：-128 至 127，用于有符号配置值)
- 3: u16 (16 位无符号整数，范围：0-65535，例如：GPIO 引脚编号)
- 4: i16 (16 位有符号整数，范围：-32768 至 32767)
- 5: u32 (32 位无符号整数，范围：0-4294967295)

- 6: i32 (32 位有符号整数, 范围: -2147483648 至 2147483647, 例如: UART 波特率 115200)
- 7: string (文本字符串, 例如: 模组名称 “MINI-1”、国家代码 “CN”、低功耗蓝牙服务配置)
- 8: binary (二进制数据, 例如: 证书、私钥)

2.2.1.3 其他重要说明

- 长度参数说明: 响应中的 <length> 参数含义因数据类型而异:  
对于数值类型 (类型 1-6) : <length> 表示该数值的字符串表示的字符数。例如, i32 类型的波特率值 115200 显示为 6 个字符, 因此 length 为 6。  
对于字符串和二进制类型 (类型 7-8) : <length> 表示实际的字节数。
- 二进制数据传输: 写入二进制数据 (类型 8) 时, 命令返回 > 并等待写入 <value> 参数中指定的确切字节数。
- 数据持久化: 写入 manufacturing NVS 的数据在设备重启后仍然保留, 直到被明确擦除。
- 偏移读取: 可以使用偏移和长度参数从键中读取部分数据, 这对处理大数据块很有用。
- 编译时与运行时: 虽然某些数据 (如低功耗蓝牙服务) 可以在编译时通过修改源 CSV 文件并重新生成 mfg\_nvs.bin 进行自定义, 但 AT+SYSMFG 命令允许在运行时更新某些值, 无需重新编译。
- 参考资料: 有关非易失性存储 (NVS) 概念和其他 NVS 操作的更多信息, 请参考 ESP-IDF NVS 文档。

2.2.2 出厂参数配置

factory\_param 命名空间存储设备的基本出厂参数, 如 Wi-Fi 配置、UART 配置和模组信息。XY-MWC320A 在初始化过程中会读取这些参数, 它们影响设备的默认运行行为。详情请参考 出厂参数配置介绍。

factory\_param 命名空间中预定义的键名及说明:

| 键名            | 说明                                                   | 详见        |
|---------------|------------------------------------------------------|-----------|
| module_name   | 模组名称, 用于标识设备型号。                                      | 模组名称操作    |
| max_tx_power  | Wi-Fi 最大发射功率, 取值范围参见 AT+RFPOWER 命令的 <wifi_power> 参数。 | 最大发射功率操作  |
| country_code  | Wi-Fi 国家代码, 相关命令: AT+CWCOUNTRY。                      | 国家代码操作    |
| start_channel | Wi-Fi 起始信道。                                          | 国家代码操作    |
| channel_num   | Wi-Fi 信道数量。                                          | 国家代码操作    |
| uart_port     | UART 端口号。                                            | UART 配置操作 |
| uart_baudrate | UART 波特率 (如 115200)。                                 | UART 配置操作 |

|              |                                                 |           |
|--------------|-------------------------------------------------|-----------|
| uart_tx_pin  | UART TX 引脚编号。                                   | UART 配置操作 |
| uart_rx_pin  | UART RX 引脚编号。                                   | UART 配置操作 |
| uart_cts_pin | UART CTS 引脚编号。                                  | UART 配置操作 |
| uart_rts_pin | UART RTS 引脚编号。                                  | UART 配置操作 |
| sys_store    | 系统存储模式，控制是否将配置自动保存到 flash，相关命令：<br>AT+SYSSTORE。 | 系统存储模式操作  |

除上表列出的预定义键名外，也可以在 factory\_param 命名空间中 添加自定义键值对 来存储应用特定的配置数据。

重要

修改出厂参数后，执行 AT+RST 或 AT+RESTORE 命令以重启设备，使更改生效。

2.2.2.1 查询所有出厂参数

```
AT
1 AT+SYSMFG=1,"factory_param"

Plain Text
1 +SYSMFG:"factory_param","version",2
2 +SYSMFG:"factory_param","module_name",7
3 +SYSMFG:"factory_param","max_tx_power",2
4 +SYSMFG:"factory_param","country_code",7
5 +SYSMFG:"factory_param","start_channel",1
6 +SYSMFG:"factory_param","channel_num",1
7 +SYSMFG:"factory_param","uart_port",2
8 +SYSMFG:"factory_param","uart_baudrate",6
9 +SYSMFG:"factory_param","uart_tx_pin",6
10 +SYSMFG:"factory_param","uart_rx_pin",6
11 +SYSMFG:"factory_param","uart_cts_pin",6
12 +SYSMFG:"factory_param","uart_rts_pin",6
13 +SYSMFG:"factory_param","sys_store",1
14 OK
```

2.2.2.2 模组名称操作

1. 读取模组名称：

```
AT
1 AT+SYSMFG=1,"factory_param","module_name"
```

Plain Text

```
1 +SYSMFG:"factory_param","module_name",7,6,MINI-1
2 OK
```

### 1. 写入新的模组名称:

AT

```
1 AT+SYSMFG=2,"factory_param","module_name",7,<name_length>
```

Plain Text

```
1 OK
```

>

返回 > 后, 发送模组名称字符串。例如, 发送 “XY-MWC320A” 。

数据发送后的响应:

Plain Text

```
1 OK
```

### 1. 验证配置是否生效:

AT

```
1 AT+SYSMFG=1,"factory_param","module_name"
```

Plain Text

```
1 +SYSMFG:"factory_param","module_name",7,10,XY-MWC320A
2 OK
```

模组名称已成功更新为 “XY-MWC320A” 。

## 2.2.2.3 UART 配置操作

### 1. 读取 UART 端口:

AT

```
1 AT+SYSMFG=1,"factory_param","uart_port"
```

Plain Text

```
1 +SYSMFG:"factory_param","uart_port",2,1,1
2 OK
```

### 1. 读取 UART 波特率:

AT

```
1 AT+SYSMFG=1,"factory_param","uart_baudrate"
```

Plain Text

```
1 +SYSMFG:"factory_param","uart_baudrate",6,6,115200
2 OK
```

### 1. 更新 UART TX 引脚:

AT

```
1 AT+SYSMFG=2,"factory_param","uart_tx_pin",6,17
```

Plain Text

```
1 OK
```

### 备注

重启后，需要切换硬件连接到新引脚或通过新的 UART 配置发送 AT 命令来验证修改是否生效。

## 2.2.2.4 最大发射功率操作

### 1. 读取最大发射功率:

AT

```
1 AT+SYSMFG=1,"factory_param","max_tx_power"
```

Plain Text

```
1 +SYSMFG:"factory_param","max_tx_power",2,2,78
2 OK
```

### 1. 更新最大发射功率:

AT

```
1 AT+SYSMFG=2,"factory_param","max_tx_power",2,60
```

Plain Text

```
1 OK
```

关于可以设置的 Wi-Fi 最大发射功率值，请参考 AT+RFPOWER 命令中 <wifi\_power> 参数的取值范围。

### 1. 重启设备使配置生效:

AT

```
1 AT+RST
```

Plain Text

```
1 ready
```

### 1. 验证配置是否生效:

AT

```
1 AT+SYSMFG=1,"factory_param","max_tx_power"
```

Plain Text

```
1 +SYSMFG:"factory_param","max_tx_power",2,2,60
2 OK
```

最大发射功率已成功更新。

#### 1. 使用 AT+RFPOWER 命令进一步验证：

AT

```
1 AT+RFPOWER?
```

Plain Text

```
1 +RFPOWER:60
2 OK
```

最大发射功率现在为 60 ( $60 \times 0.25 \text{ dBm} = 15 \text{ dBm}$ )。

### 2.2.2.5 国家代码操作

#### 1. 读取国家代码：

AT

```
1 AT+SYSMFG=1,"factory_param","country_code"
```

Plain Text

```
1 +SYSMFG:"factory_param","country_code",7,2,CN
2 OK
```

#### 1. 更新国家代码：

AT

```
1 AT+SYSMFG=2,"factory_param","country_code",7,2
```

Plain Text

```
1 OK
```

>

收到 > 后，发送国家代码（例如 “US”、“JP”、“EU”）。

数据传输后的响应：

Plain Text

```
1 OK
```

#### 1. 重启设备使配置生效：

AT

1 AT+RST

Plain Text

1 ready

### 1. 验证配置是否生效:

AT

1 AT+SYSMFG=1,"factory\_param","country\_code"

Plain Text

1 +SYSMFG:"factory\_param","country\_code",7,2,US

2 OK

国家代码已成功更新为 “US” 。

### 1. 更新信道数量为 11（美国标准）：

AT

1 AT+SYSMFG=2,"factory\_param","channel\_num",1,11

Plain Text

1 OK

### 1. 重启设备使配置生效:

AT

1 AT+RST

Plain Text

1 ready

### 1. 使用 AT+CWCOUNTRY 命令验证国家码和信道配置:

AT

1 AT+CWCOUNTRY?

Plain Text

1 +CWCOUNTRY:0,"US",1,11

2 OK

响应说明:

- 0: 国家码策略 (0 表示使用默认国家码配置)
- "US": 国家代码

- 1: 起始信道 (start\_channel)
- 11: 信道数量 (channel\_num)

配置已成功生效，设备现在使用美国的国家代码和信道配置。

### 2.2.2.6 系统存储模式操作

sys\_store 参数控制系统是否将配置更改自动存储到 flash。此参数与 AT+SYSSTORE 命令的功能相关。

#### 1. 读取当前系统存储模式：

AT

```
1 AT+SYSMFG=1,"factory_param","sys_store"
```

Plain Text

```
1 +SYSMFG:"factory_param","sys_store",1,1,1
2 OK
```

sys\_store 的值：

- 0: 不自动存储配置到 flash (相当于 AT+SYSSTORE=0)
- 1: 自动存储配置到 flash (相当于 AT+SYSSTORE=1, 默认值)

#### 1. 更改系统存储模式为不自动存储：

AT

```
1 AT+SYSMFG=2,"factory_param","sys_store",1,0
```

Plain Text

```
1 OK
```

#### 1. 重启设备使配置生效：

AT

```
1 AT+RST
```

Plain Text

```
1 ready
```

#### 1. 验证配置是否生效：

AT

```
1 AT+SYSMFG=1,"factory_param","sys_store"
```

Plain Text

```
1 +SYSMFG:"factory_param","sys_store",1,1,0
2 OK
```

## 1. 使用 AT+SYSSTORE 命令进一步验证:

AT

1 AT+SYSSTORE?

Plain Text

1 +SYSSTORE:0

2 OK

系统存储模式已成功更改为不自动存储（0 表示关闭自动存储）。

更改后的影响:

当 sys\_store 设置为 0 后，配置更改（如 Wi-Fi 设置或其他支持自动存储的命令）不会自动保存到 flash，重启后会恢复为之前保存的配置。受影响的命令详见 AT+SYSSTORE。

如果需要在 sys\_store=0 模式下保存特定配置，可以使用 AT+SYSSTORE 命令临时启用存储，执行配置命令后再关闭。

### 2.2.2.7 添加自定义键值对

除了预定义的出厂参数外，还可以在 factory\_param 命名空间中添加自定义键值对来存储应用特定的配置数据。

本示例演示如何添加一个自定义键值对，例如存储设备序列号:

#### 1. 添加自定义字符串类型键值对:

AT

1 AT+SYSMFG=2,"factory\_param","device\_sn",7,16

Plain Text

1 OK

>

返回 > 后，发送设备序列号（例如 “SN202601050001”）。

数据发送后的响应:

Plain Text

1 OK

#### 1. 读取自定义键值对:

AT

1 AT+SYSMFG=1,"factory\_param","device\_sn"

Plain Text

```
1 +SYSMFG:"factory_param","device_sn",7,16,SN202601050001
2 OK
```

### 1. 添加数值类型的自定义键值对：

AT

```
1 AT+SYSMFG=2,"factory_param","hw_version",6,1001
```

Plain Text

```
1 OK
```

这将存储硬件版本号 1001。

### 1. 查询所有键值对（包括自定义的）：

AT

```
1 AT+SYSMFG=1,"factory_param"
```

Plain Text

```
1 +SYSMFG:"factory_param","version",2
2 +SYSMFG:"factory_param","module_name",7
```

...

Plain Text

```
1 +SYSMFG:"factory_param","device_sn",7
2 +SYSMFG:"factory_param","hw_version",6
3 OK
```

### 1. 删除自定义键值对：

AT

```
1 AT+SYSMFG=0,"factory_param","device_sn"
```

Plain Text

```
1 OK
```

### 备注

- 自定义键名应避免与预定义的键名冲突（如 module\_name、uart\_port 等）
- 建议使用有意义的键名，便于后续维护
- 自定义键值对在重启后保持持久化
- 过多的键值对可能影响性能，建议仅存储必要的配置信息
- 修改 factory\_param 中的值后，建议重启设备以确保应用程序正确读取新值

## 2.2.3 PKI 配置

PKI（公钥基础设施）配置用于创建、管理、分配、存储和撤销数字证书，以及管理公钥加密。XY-MWC320A 支持为多种功能配置证书，包括 SSL 客户端/服务器、MQTT 客户端、HTTPS 客户端等。所有证书的配置方法完全相同，均通过 AT+SYSMFG 命令操作对应的命名空间实现。更多详情请参考 PKI 配置介绍。

不同功能的命名空间和键名映射：

| 功能        | 命名空间                                   | 键名                                                                                                 | 说明                                                                                                                                                                                                                           |
|-----------|----------------------------------------|----------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SSL 客户端   | client_cert<br>client_key<br>client_ca | client_cert.0 ,<br>client_cert.1<br>client_key.0 ,<br>client_key.1<br>client_ca.0 ,<br>client_ca.1 | AT 作为 SSL 客户端时，需要配置客户端证书，提供给服务器进行身份验证。按 client_cert.<index> 规则存储多套证书。AT 作为 SSL 客户端时，需要配置客户端私钥，用于加密客户端与服务器之间的通信。按 client_key.<index> 规则存储多套私钥。AT 作为 SSL 客户端时，需要配置客户端 CA 证书，用于校验服务器证书的有效性。按 client_ca.<index> 规则存储多套 CA 证书。    |
| SSL 服务器   | server_cert<br>server_key<br>server_ca | server_cert<br>server_key<br>server_ca                                                             | AT 作为 SSL 服务器时，需要配置服务器证书，用于验证客户端的身份。只支持单套证书。AT 作为 SSL 服务器时，需要配置服务器私钥，用于加密服务器与客户端之间的通信。只支持单套私钥。AT 作为 SSL 服务器时，需要配置服务器 CA 证书用于校验客户端证书的有效性。只支持单套 CA 证书。                                                                         |
| MQTT 客户端  | mqtt_cert<br>mqtt_key<br>mqtt_ca       | mqtt_cert<br>mqtt_key<br>mqtt_ca                                                                   | AT 作为 MQTT 客户端时，需要配置客户端证书，提供给服务器进行身份验证。只支持单套证书。AT 作为 MQTT 客户端时，需要配置客户端私钥，用于加密客户端与服务器之间的通信。只支持单套私钥。AT 作为 MQTT 客户端时，需要配置客户端 CA 证书，用于校验服务器证书的有效性。只支持单套 CA 证书。                                                                   |
| HTTPS 客户端 | https_cert<br>https_key<br>https_ca    | https_cert.0 ,<br>https_cert.1<br>https_key.0 ,<br>https_key.1<br>https_ca.0 ,<br>https_ca.1       | AT 作为 HTTPS 客户端时，需要配置客户端证书，提供给服务器进行身份验证。按 https_cert.<index> 规则存储多套证书。AT 作为 HTTPS 客户端时，需要配置客户端私钥，用于加密客户端与服务器之间的通信。按 https_key.<index> 规则存储多套私钥。AT 作为 HTTPS 客户端时，需要配置客户端 CA 证书，用于校验服务器证书的有效性。按 https_ca.<index> 规则存储多套 CA 证书。 |

|                                                        |                                           |                                                                                  |                                                                                                                                                                                                                                                                                                                                      |
|--------------------------------------------------------|-------------------------------------------|----------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| WebSocket 客户端                                          | wss_cert<br>wss_key<br>wss_ca             | wss_cert.0 ,<br>wss_cert.1<br>wss_key.0 ,<br>wss_key.1<br>wss_ca.0 ,<br>wss_ca.1 | AT 作为 WebSocket 客户端时，需要配置客户端证书，提供给服务器进行身份验证。按 wss_cert.<index> 规则存储多套证书。AT 作为 WebSocket 客户端时，需要配置客户端私钥，用于加密客户端与服务器之间的通信。按 wss_key.<index> 规则存储多套私钥。AT 作为 WebSocket 客户端时，需要配置客户端 CA 证书，用于校验服务器证书的有效性。按 wss_ca.<index> 规则存储多套 CA 证书。                                                                                                   |
| Wi-Fi Enterprise 客户端 (WPA2-Enterprise/WPA3-Enterprise) | wpa_ent_cert<br>wpa_ent_key<br>wpa_ent_ca | wpa_ent_cert<br>wpa_ent_key<br>wpa_ent_ca                                        | AT 作为 Wi-Fi Enterprise 客户端时，需要配置客户端证书，提供给服务器进行身份验证。只支持单套证书。AT 作为 Wi-Fi Enterprise 客户端时，需要配置客户端私钥，用于加密客户端与服务器之间的通信。只支持单套私钥。AT 作为 Wi-Fi Enterprise 客户端时，需要配置客户端 CA 证书，用于校验服务器证书的有效性。只支持单套 CA 证书。上述命名空间和键名由 WPA2-Enterprise 与 WPA3-Enterprise 共用。若使能 CONFIG_AT_EAP_LEGACY_NAMESPACE_SUPPORT，则会改用旧的 wpa2_cert、wpa2_key、wpa2_ca 命名以兼容旧固件。 |

备注

可以通过 AT+SYSMFG? 命令查询当前固件支持的所有命名空间，以确认需要的证书命名空间是否可用。本节以 SSL 证书为例说明证书配置的完整流程。其他功能的证书配置（如 MQTT、HTTPS）可参照相同方法，仅需根据上表修改对应的命名空间名称和键名即可。

2.2.3.1 SSL 客户端证书

client\_cert、client\_key 和 client\_ca 命名空间存储用于安全连接的 SSL 证书和密钥。这三个命名空间可以协同工作以实现 SSL 认证（单向或双向认证）。

2.2.3.1.1 查询证书信息

1. 读取客户端证书命名空间中的所有键值对：

AT

1 AT+SYSMFG=1,"client\_cert"

Plain Text

1 +SYSMFG:"client\_cert","client\_cert.0",8

2 +SYSMFG:"client\_cert","client\_cert.1",8

3 OK

同样，可以查询 client\_key 和 client\_ca 命名空间：

AT

- 1 AT+SYSMFG=1,"client\_key"
- 2 AT+SYSMFG=1,"client\_ca"

#### 1. 读取特定的证书或密钥：

AT

- 1 AT+SYSMFG=1,"client\_cert","client\_cert.0"

Plain Text

- 1 +SYSMFG:"client\_cert","client\_cert.0",8,1164,<value>
- 2 OK

### 2.2.3.1.2 配置证书

#### 1. 写入客户端证书：

AT

- 1 AT+SYSMFG=2,"client\_cert","client\_cert.0",8,1164

Plain Text

- 1 OK

>

返回 > 后，发送证书数据（1164 字节）。数据发送完成后，系统返回：

Plain Text

- 1 OK

#### 1. 写入客户端密钥：

AT

- 1 AT+SYSMFG=2,"client\_key","client\_key.0",8,<key\_length>

Plain Text

- 1 OK

>

返回 > 后，发送密钥数据。数据发送完成后，系统返回：

Plain Text

- 1 OK

#### 1. 写入 CA 证书：

AT

```
1 AT+SYSMFG=2,"client_ca","client_ca.0",8,<ca_length>
```

Plain Text

```
1 OK
```

>

返回 > 后，发送 CA 证书数据。数据发送完成后，系统返回：

Plain Text

```
1 OK
```

### 2.2.3.1.3 使用证书

将客户端证书、密钥（双向认证时还需客户端 CA 证书）写入 manufacturing NVS 分区后，可以使用它们建立安全的 SSL 连接。以下是示例：

#### 1. 确保证书已写入 manufacturing 分区：

客户端证书存储在 client\_cert 命名空间

客户端密钥存储在 client\_key 命名空间

客户端 CA 证书存储在 client\_ca 命名空间

#### 2. 配置 SSL 连接使用 manufacturing 分区证书：

AT

```
1 AT+CIPSSLCONF=3,0,0
```

Plain Text

```
1 OK
```

这将配置 SSL 客户端使用 manufacturing 分区中的证书（索引为 0）。

#### 3. 连接到 Wi-Fi：

AT

```
1 AT+CWJAP="your_ssid","your_password"
```

Plain Text

```
1 OK
```

#### 4. 建立到服务器的 SSL 连接：

AT

```
1 AT+CIPSTART="SSL","example.com",8883
```

Plain Text

- 1 CONNECT
- 2 OK

SSL 连接将使用 manufacturing 分区中的证书进行身份验证。

## 备注

有关 SSL 配置的更多详细信息，请参阅 AT+CIPSSLCONF 命令文档。

### 2.2.3.1.4 擦除证书

擦除特定的证书：

AT

- 1 AT+SYSMFG=0,"client\_cert","client\_cert.0"

Plain Text

- 1 OK

擦除命名空间中的所有键值对：

AT

- 1 AT+SYSMFG=0,"client\_cert"

Plain Text

- 1 OK

可以使用相同的命令擦除 client\_key 和 client\_ca 数据。

### 2.2.3.2 SSL 服务器证书

server\_cert、server\_key 和 server\_ca 命名空间存储设备作为 SSL 服务器所需的证书和密钥。这三个命名空间可以协同工作以启用安全的服务器端 SSL 连接（单向或双向认证）。

#### 2.2.3.2.1 查询证书信息

1. 读取服务器证书命名空间中的所有键：

AT

- 1 AT+SYSMFG=1,"server\_cert"

Plain Text

- 1 +SYSMFG:"server\_cert","server\_cert.0",8
- 2 +SYSMFG:"server\_cert","server\_cert.1",8
- 3 OK

同样，可以查询 server\_key 和 server\_ca 命名空间：

AT

- 1 AT+SYSMFG=1,"server\_key"
- 2 AT+SYSMFG=1,"server\_ca"

### 1. 读取特定的服务器证书或密钥:

AT

- 1 AT+SYSMFG=1,"server\_cert","server\_cert.0"

Plain Text

- 1 +SYSMFG:"server\_cert","server\_cert.0",8,<length>,<value>
- 2 OK

AT

- 1 AT+SYSMFG=1,"server\_key","server\_key.0"

Plain Text

- 1 +SYSMFG:"server\_key","server\_key.0",8,<length>,<value>
- 2 OK

## 2.2.3.2.2 配置证书

### 1. 写入服务器证书:

AT

- 1 AT+SYSMFG=2,"server\_cert","server\_cert.0",8,<cert\_length>

Plain Text

- 1 OK

>

返回 > 后, 发送证书数据。

数据传输后的响应:

Plain Text

- 1 OK

### 1. 写入服务器密钥:

AT

- 1 AT+SYSMFG=2,"server\_key","server\_key.0",8,<key\_length>

Plain Text

- 1 OK

>

返回 > 后，发送密钥数据。

数据传输后的响应：

Plain Text

1 OK

### 1. 写入 CA 证书：

AT

1 AT+SYSMFG=2,"server\_ca","server\_ca.0",8,<ca\_length>

Plain Text

1 OK

>

返回 > 后，发送 CA 证书数据。

数据传输后的响应：

Plain Text

1 OK

## 2.2.3.2.3 使用证书

将服务器证书、密钥（双向认证时还需服务器 CA 证书）写入 manufacturing NVS 分区后，可以基于这些证书创建 SSL 服务器。以下是一个完整的示例：

### 1. 确保服务器凭证已写入 manufacturing 分区：

服务器证书位于 server\_cert 命名空间

服务器密钥位于 server\_key 命名空间

服务器 CA 证书位于 server\_ca 命名空间（可选，仅在需要客户端认证时使用）

### 2. 启用多连接模式：

AT

1 AT+CIPMUX=1

Plain Text

1 OK

### 3. 启动 SSL 服务器：

如果只需要服务器单向认证（服务器向客户端提供证书）：

AT

1 AT+CIPSERVER=1,8883,"SSL"

Plain Text

1 OK

如果需要双向认证（服务器和客户端相互验证证书）：

AT

1 AT+CIPSERVER=1,8883,"SSL",1

Plain Text

1 OK

其中最后一个参数 1 表示启用 CA 验证，服务器将使用 manufacturing 分区中的 CA 证书验证客户端证书。

设备现在在端口 8883 上监听 SSL 连接，SSL 服务器将自动使用 manufacturing 分区中的服务器证书和密钥。

- 如果只需要服务器单向认证（服务器向客户端提供证书）：

AT

1 AT+CIPSERVER=1,8883,"SSL"

Plain Text

1 OK

- 如果需要双向认证（服务器和客户端相互验证证书）：

AT

1 AT+CIPSERVER=1,8883,"SSL",1

Plain Text

1 OK

其中最后一个参数 1 表示启用 CA 验证，服务器将使用 manufacturing 分区中的 CA 证书验证客户端证书。

1. 当客户端连接时，SSL 握手将使用 manufacturing 分区中的服务器证书和密钥进行身份验证。

#### 备注

- SSL 服务器会自动从 manufacturing 分区 (manufacturing partition) 读取 server\_cert 和 server\_key 命名空间中的证书和密钥，无需额外配置。
- 如果在 AT+CIPSERVER 命令中启用了 CA 验证 (<CA enable> = 1)，服务器还会从 server\_ca 命名空间读取 CA 证书来验证客户端证书，实现双向认证。
- 有关 SSL 服务器配置的更多详细信息，请参阅 AT+CIPSERVER 命令文档。

### 2.2.3.2.4 擦除证书

擦除特定的服务器证书：

```
AT
1 AT+SYSMFG=0,"server_cert","server_cert.0"

Plain Text
1 OK
```

擦除命名空间中的所有键：

```
AT
1 AT+SYSMFG=0,"server_cert"

Plain Text
1 OK
```

可以使用相同的命令擦除 server\_key 和 server\_ca 数据。

### 2.2.4 GATTS 配置

ble\_data 命名空间存储低功耗蓝牙 GATTS (Generic Attribute Profile Server) 服务配置。该命名空间包含以 cfg0、cfg1、cfg2 等形式组织的各个 GATT 服务配置项，其中每个配置项表示 GATT 表中的一个属性（服务、特征或描述符）。更多详情请参考 低功耗蓝牙服务源文件介绍。

ble\_data 命名空间中的键名及说明：

| 键名                          | 说明                                                 | 相关 AT 命令          |
|-----------------------------|----------------------------------------------------|-------------------|
| cfg0 , cfg1 , ...,<br>cfg30 | 低功耗蓝牙服务配置项，每个 cfg<index> 对应 GATT 表中的一行（服务、特征或描述符）。 | AT+BLEGATTSSRVCRE |

#### 重要

- 使用 AT+SYSMFG 命令修改 ble\_data 命名空间时，需要在未初始化 BLE 的前提下执行。如果 BLE 已初始化，请先执行 AT+BLEINIT=0 关闭 BLE。
- 在使用 AT+SYSMFG 读取或写入 ble\_data 命名空间之前，请先阅读 低功耗蓝牙服务源文件介绍 文档，了解低功耗蓝牙服务配置的结构和格式。

#### 2.2.4.1 查询配置信息

##### 2.2.4.1.1 读取配置项

1. 读取 ble\_data 命名空间中的所有配置项：

```
AT
1 AT+SYSMFG=1,"ble_data"
```

Plain Text

```
1 +SYSMFG:"ble_data","cfg0",7
2 +SYSMFG:"ble_data","cfg1",7
3 +SYSMFG:"ble_data","cfg2",7
4 +SYSMFG:"ble_data","cfg3",7
```

...

Plain Text

```
1 +SYSMFG:"ble_data","cfg30",7
2 OK
```

1. 读取特定的 BLE 配置项（例如 cfg2 表示 GATT 服务表中的一行）：

AT

```
1 AT+SYSMFG=1,"ble_data","cfg2"
```

Plain Text

```
1 +SYSMFG:"ble_data","cfg2",7,23,2,16,0xC300,0x01,1,1,30
2 OK
```

响应格式：

响应数据表示 GATT 表中的以下字段：

- 2: index (索引)
- 16: uuid\_len (UUID 长度, 以位为单位)
- 0xC300: uuid (UUID)
- 0x01: perm (权限)
- 1: val\_max\_len (最大值长度)
- 1: val\_cur\_len (当前值长度)
- 30: value (值)

## 2.2.4.2 配置服务

### 2.2.4.2.1 修改现有 BLE 特征的 UUID

此示例展示如何在运行时将特征的 UUID 从 0xC300 修改为 0xC500。

1. 查询当前配置：

AT

```
1 AT+SYSMFG=1,"ble_data","cfg2"
```

Plain Text

```
1 +SYSMFG:"ble_data","cfg2",7,23,2,16,0xC300,0x01,1,1,30
2 OK
```

当前 UUID 为 0xC300。

### 1. 写入 UUID 为 0xC500 的新配置:

AT

```
1 AT+SYSMFG=2,"ble_data","cfg2",7,23
```

Plain Text

```
1 OK
```

>

发送数据:

AT

```
1 2,16,0xC500,0x01,1,1,30
```

Plain Text

```
1 OK
```

### 1. 查询配置项验证 UUID 是否修改成功:

AT

```
1 AT+SYSMFG=1,"ble_data","cfg2"
```

Plain Text

```
1 +SYSMFG:"ble_data","cfg2",7,23,2,16,0xC500,0x01,1,1,30
2 OK
```

UUID 已成功更改为 0xC500。

### 1. 重新初始化低功耗蓝牙并创建服务以应用更改:

AT

```
1 AT+BLEINIT=2
```

Plain Text

```
1 OK
```

AT

```
1 AT+BLEGATTSSRVCRE
```

Plain Text

```
1 OK
```

AT

1 AT+BLEGATTSSRVSTART

Plain Text

1 OK

### 1. 查询服务以验证:

AT

1 AT+BLEGATTSSRV?

Plain Text

1 +BLEGATTSSRV:1,1,0xA002,1  
2 +BLEGATTSSRV:1,2,0xA003,1  
3 OK

### 1. 查询特征以验证 UUID 更改:

AT

1 AT+BLEGATTSSCHAR?

Plain Text

1 +BLEGATTSSCHAR:"char",1,1,0xC500  
2 +BLEGATTSSCHAR:"char",1,2,0xC301  
3 +BLEGATTSSCHAR:"char",1,3,0xC302  
4 +BLEGATTSSCHAR:"char",1,4,0xC303  
5 +BLEGATTSSCHAR:"char",1,5,0xC304  
6 +BLEGATTSSCHAR:"char",1,6,0xC305  
7 +BLEGATTSSCHAR:"char",1,7,0xC306  
8 +BLEGATTSSCHAR:"char",1,8,0xC307  
9 +BLEGATTSSCHAR:"char",2,1,0xC400  
10 +BLEGATTSSCHAR:"char",2,2,0xC401  
11 OK

第一个特征现在显示 UUID 为 0xC500 而不是 0xC300。

#### 2.2.4.2.2 添加新的自定义 BLE 服务

此示例展示如何在运行时添加一系列新的 BLE 服务，而不修改现有的服务配置。假设需要添加一个自定义服务（UUID: 0xA100），包含一个可读写的特征（UUID: 0xB100）。

### 1. 查询当前最后一个配置项的索引:

AT

1 AT+SYSMFG=1,"ble\_data"

响应示例:

Plain Text

```
1 +SYSMFG:"ble_data","cfg0",7
2 +SYSMFG:"ble_data","cfg1",7
```

...

Plain Text

```
1 +SYSMFG:"ble_data","cfg30",7
2 OK
```

假设当前最后一个配置项是 cfg30，此处从 cfg31 开始添加新服务。

### 1. 添加主服务声明 (Primary Service Declaration):

AT

```
1 AT+SYSMFG=2,"ble_data","cfg31",7,26
```

Plain Text

```
1 OK
```

>

发送数据:

AT

```
1 31,16,0x2800,0x01,2,2,A100
```

数据格式说明:

- 31: 配置项索引
- 16: UUID 长度 (16 位)
- 0x2800: Primary Service Declaration UUID
- 0x01: 权限 (可读)
- 2: 最大值长度 (2 字节, 用于存储服务 UUID)
- 2: 当前值长度
- A100: 自定义服务的 UUID

Plain Text

```
1 OK
```

### 1. 添加特征声明 (Characteristic Declaration):

AT

```
1 AT+SYSMFG=2,"ble_data","cfg32",7,24
```

Plain Text

1 OK

>

发送数据:

AT

1 32,16,0x2803,0x01,3,1,02

数据格式说明:

- 32: 配置项索引
- 16: UUID 长度
- 0x2803: 特征声明
- 0x01: 权限 (可读)
- 3: 最大值长度
- 1: 当前值长度
- 02: 设置权限为读

Plain Text

1 OK

### 1. 添加特征值 (Characteristic Value):

AT

1 AT+SYSMFG=2,"ble\_data","cfg33",7,31

Plain Text

1 OK

>

发送数据:

AT

1 33,16,0xB100,0x11,20,8,12345678

数据格式说明:

- 33: 配置项索引
- 16: UUID 长度
- 0xB100: 特征值的 UUID (与特征声明中的 UUID 一致)

- 0x11: 权限 (读+写: ESP\_GATT\_PERM\_READ (0x01) + ESP\_GATT\_PERM\_WRITE (0x10) = 0x11)
- 20: 最大值长度 (允许最多 20 字节的数据)
- 8: 当前值长度 (初始值为 8 字节)
- 12345678: 初始值 (8 字节数据)

Plain Text

1 OK

### 1. 查询新添加的配置项验证:

AT

1 AT+SYSMFG=1,"ble\_data","cfg31"

Plain Text

1 +SYSMFG:"ble\_data","cfg31",7,26,31,16,0x2800,0x01,2,2,A100

2 OK

AT

1 AT+SYSMFG=1,"ble\_data","cfg32"

### 响应示例:

Plain Text

1 +SYSMFG:"ble\_data","cfg32",7,24,32,16,0x2803,0x01,3,1,02

2 OK

AT

1 AT+SYSMFG=1,"ble\_data","cfg33"

### 响应示例:

Plain Text

1 +SYSMFG:"ble\_data","cfg33",7,31,33,16,0xB100,0x11,20,8,12345678

2 OK

### 1. 重新初始化低功耗蓝牙并创建服务以应用更改:

AT

1 AT+BLEINIT=2

Plain Text

1 OK

AT

1 AT+BLEGATTSSRVCRE

Plain Text

1 OK

AT

1 AT+BLEGATTSSRVSTART

Plain Text

1 OK

### 1. 查询服务验证新服务已添加：

AT

1 AT+BLEGATTSSRV?

响应示例：

Plain Text

1 +BLEGATTSSRV:1,1,0xA002,1

2 +BLEGATTSSRV:1,2,0xA003,1

3 +BLEGATTSSRV:1,3,0xA100,1

4 OK

新添加的服务 UUID 0xA100 已出现在服务列表中。

### 1. 查询特征验证新特征已添加：

AT

1 AT+BLEGATTSSCHAR?

响应示例：

Plain Text

1 +BLEGATTSSCHAR:"char",1,1,0xC300

2 +BLEGATTSSCHAR:"char",1,2,0xC301

...

Plain Text

1 +BLEGATTSSCHAR:"char",3,1,0xB100

2 OK

新添加的特征 UUID 0xB100 已出现在特征列表中（属于服务 3，特征索引 1）。

备注

BLE 服务配置要点：

- 添加新服务时，确保使用的 cfg 索引不与现有配置冲突
- 一个完整的 BLE 服务至少需要三个配置项：主服务声明、特征声明、特征值
- 如需特征描述符（如 Client Characteristic Configuration Descriptor），需添加额外配置项
- 特征属性值可组合使用，常用属性包括：0x01（广播）、0x02（读）、0x04（无响应写）、0x08（写）、0x10（通知）、0x20（指示）、0x40（认证签名写）、0x80（扩展属性）

数据格式与生效：

- 每个 cfg 项对应 gatts\_data.csv 文件中 GATT 服务表的一行
- 数据格式：index,uuid\_len,uuid,perm,val\_max\_len,val\_cur\_len,value
- 通过 AT+SYSMFG 写入的配置需重新初始化 BLE（AT+BLEINIT）并重新创建服务（AT+BLEGATTSSRVCRE）后生效
- 更多 GATT 表结构详情，请参阅 如何自定义低功耗蓝牙服务

## 2.2.5 自定义数据存储

除了预定义的命名空间（如 factory\_param、client\_cert、server\_cert 等）之外，还可以创建自定义命名空间来存储自己的应用数据。这对于存储需要在重启后保留的少量持久化数据非常有用，例如自定义配置、用户偏好设置或设备特定的参数。

### 2.2.5.1 创建和使用自定义命名空间

本示例演示如何创建一个名为 my\_app\_ca 的自定义命名空间来存储应用特定的证书或配置数据。

#### 1. 创建新命名空间并写入数据：

```
AT
1 AT+SYSMFG=2,"my_app_ca","my_app_ca.0",8,100
```

```
Plain Text
1 OK
```

>

返回 > 后，发送的二进制数据（100 字节）。数据发送完成后，系统返回：

```
Plain Text
1 OK
```

命名空间 my\_app\_ca 现已创建，其中包含一个键 my\_app\_ca.0，存储了相应数据。

#### 1. 查询自定义命名空间：

```
AT
1 AT+SYSMFG=1,"my_app_ca"
```

Plain Text

```
1 +SYSMFG:"my_app_ca","my_app_ca.0",8
2 OK
```

### 1. 从自定义命名空间读取数据:

AT

```
1 AT+SYSMFG=1,"my_app_ca","my_app_ca.0"
```

Plain Text

```
1 +SYSMFG:"my_app_ca","my_app_ca.0",8,100,<data>
2 OK
```

响应包含数据类型（8 表示二进制）、长度（100 字节）和实际数据。

### 1. 向同一命名空间添加更多键:

AT

```
1 AT+SYSMFG=2,"my_app_ca","my_app_ca.1",7,20
```

Plain Text

```
1 OK
```

>

发送数据:

MyCustomConfigValue1

Plain Text

```
1 OK
```

### 1. 查询命名空间中的所有键:

AT

```
1 AT+SYSMFG=1,"my_app_ca"
```

Plain Text

```
1 +SYSMFG:"my_app_ca","my_app_ca.0",8
2 +SYSMFG:"my_app_ca","my_app_ca.1",7
3 OK
```

### 1. 创建另一个命名空间存储应用配置参数:

如需存储配置参数（而非证书数据），建议创建单独的命名空间以保持语义清晰。

AT

```
1 AT+SYSMFG=2,"my_app_config","timeout",6,5000
```

Plain Text

1 OK

这将在 my\_app\_config 命名空间中存储一个 i32 类型的配置值（类型 6），键名为 “timeout”，值为 5000。

### 1. 读取配置参数：

AT

1 AT+SYSMFG=1,"my\_app\_config","timeout"

Plain Text

1 +SYSMFG:"my\_app\_config","timeout",6,4,5000

2 OK

### 1. 擦除特定的键：

AT

1 AT+SYSMFG=0,"my\_app\_ca","my\_app\_ca.1"

Plain Text

1 OK

### 1. 擦除整个命名空间：

AT

1 AT+SYSMFG=0,"my\_app\_ca"

Plain Text

1 OK

## 备注

### 自定义命名空间最佳实践：

- 自定义命名空间可用于存储任何类型的持久化数据
- 根据需求选择合适的数据类型：数值类型 (1-6) 用于数字，字符串 (7) 用于文本，二进制 (8) 用于证书或二进制数据
- 建议按用途分离命名空间：例如将证书数据 (my\_app\_ca) 与配置参数 (my\_app\_config) 分开存储，便于管理和维护
- 命名空间名称应具有描述性，清楚表明其用途（如 my\_app\_certs、my\_app\_settings）
- 键名可以是任意字符串，但建议使用有意义的名称以便识别
- 所有自定义数据在重启后保持持久化，直到被明确擦除

- 自定义命名空间独立于系统命名空间，不会影响 AT 固件功能
- 创建自定义命名空间并存储数据后，可以使用 AT+SYSMFG? 查询，在命名空间列表中看到它

## 2.3 TCP-IP AT 示例

本文档主要介绍在 XY-MWC320A 设备上运行 TCP/IP AT 命令 命令的详细示例。

### 2.3.1 XY-MWC320A 设备作为 TCP 客户端建立单连接

#### 1. 设置 Wi-Fi 模式为 station。

```
AT
1 AT+CWMODE=1
```

```
Plain Text
1 OK
```

#### 2. 连接到路由器。

```
AT
1 AT+CWJAP="XY-Router","1234567890"
```

```
Plain Text
1 WIFI CONNECTED
2 WIFI GOT IP
3 OK
```

**说明：**实际输入的 SSID 和密码可能与上述命令中的不同，请使用所用路由器的 SSID 和密码。

#### 3. 查询 XY-MWC320A 设备 IP 地址。

```
AT
1 AT+CIPSTA?
```

```
Plain Text
1 +CIPSTA:ip:"192.168.3.112"
2 +CIPSTA:gateway:"192.168.3.1"
3 +CIPSTA:netmask:"255.255.255.0"
4 OK
```

**说明：**查询结果可能与上述响应中的不同。

#### 4. PC 与 XY-MWC320A 设备连接同一个路由。

在 PC 上使用网络调试工具，创建一个 TCP 服务器。例如 TCP 服务器的 IP 地址为 192.168.3.102，端口为 8080。

5. XY-MWC320A 设备作为客户端通过 TCP 连接到 TCP 服务器，服务器 IP 地址为 192.168.3.102，端口为 8080。

AT

```
1 AT+CIPSTART="TCP","192.168.3.102",8080
```

Plain Text

```
1 CONNECT
2 OK
```

6. 发送 4 字节数据。

AT

```
1 AT+CIPSEND=4
```

Plain Text

```
1 OK
2 >
```

输入 4 字节数据，例如输入数据是 test，之后 AT 将会输出以下信息。

Plain Text

```
1 Recv 4 bytes
2 SEND OK
```

**说明：**若输入的字节数目超过 AT+CIPSEND 命令设定的长度 (n)，则系统会响应 busy p...，并发送数据的前 n 个字节，发送完成后响应 SEND OK。

7. 接收 4 字节数据。

假设 TCP 服务器发送 4 字节的数据（数据为 test），则系统会提示：

Plain Text

```
1 +IPD,4:test
```

### 2.3.2 XY-MWC320A 设备作为 TCP 服务器建立多连接

当 XY-MWC320A 设备作为 TCP 服务器时，必须通过 AT+CIPMUX=1 命令使能多连接，因为可能有多 个 TCP 客户端连接到 XY-MWC320A 设备。

以下是 XY-MWC320A 设备作为 softAP 建立 TCP 服务器的示例；如果是 XY-MWC320A 设备作为 station，可在连接路由器后按照同样方法建立服务器。

1. 设置 Wi-Fi 模式为 softAP。

AT

```
1 AT+CWMODE=2
```

Plain Text

1 OK

## 2. 使能多连接。

AT

1 AT+CIPMUX=1

Plain Text

1 OK

## 3. 设置 softAP。

AT

1 AT+CWSAP="XY\_softAP", "1234567890", 5, 3

Plain Text

1 OK

## 4. 查询 softAP 信息。

AT

1 AT+CIPAP?

Plain Text

1 AT+CIPAP?

2 +CIPAP:ip:"192.168.4.1"

3 +CIPAP:gateway:"192.168.4.1"

4 +CIPAP:netmask:"255.255.255.0"

5 OK

**说明：**查询到的地址可能与上述响应中的不同。

## 5. 建立 TCP 服务器，默认端口为 333。

AT

1 AT+CIPSERVER=1

Plain Text

1 OK

## 6. PC 连接到 XY-MWC320A 设备的 softAP。

## 7. 在 PC 上使用网络调试工具创建一个 TCP 客户端，连接到 XY-MWC320A 设备创建的 TCP 服务器。

## 8. 发送 4 字节数据到网络连接 ID 为 0 的链路上。

```
AT
1 AT+CIPSEND=0,4
```

```
Plain Text
1 OK
2 >
```

输入 4 字节数据，例如输入数据是 test，之后 AT 将会输出以下信息。

```
Plain Text
1 Recv 4 bytes
2 SEND OK
```

**说明：**若输入的字节数目超过 AT+CIPSEND 命令设定的长度 (n)，则系统会响应 busy p...，并发送数据的前 n 个字节，发送完成后响应 SEND OK。

## 9. 从网络连接 ID 为 0 的链路上接收 4 字节数据。

假设 TCP 服务器发送 4 字节的数据（数据为 test），则系统会提示：

```
Plain Text
1 +IPD,0,4:test
```

## 10. 关闭 TCP 连接。

```
AT
1 AT+CIPCLOSE=0
```

```
Plain Text
1 0,CLOSED
2 OK
```

### 2.3.3 远端 IP 地址和端口固定的 UDP 通信

#### 1. 设置 Wi-Fi 模式为 station。

```
AT
1 AT+CWMODE=1
```

```
Plain Text
1 OK
```

#### 2. 连接到路由器。

```
AT
1 AT+CWJAP="XY-Router","1234567890"
```

Plain Text

```
1 WIFI CONNECTED
2 WIFI GOT IP
3 OK
```

**说明：**实际输入的 SSID 和密码可能与上述命令中的不同，请使用所用路由器的 SSID 和密码。

### 3. 查询 XY-MWC320A 设备 IP 地址。

AT

```
1 AT+CIPSTA?
```

Plain Text

```
1 +CIPSTA:ip:"192.168.3.112"
2 +CIPSTA:gateway:"192.168.3.1"
3 +CIPSTA:netmask:"255.255.255.0"
4 OK
```

**说明：**查询结果可能与上述响应中的不同。

### 4. PC 与 XY-MWC320A 设备连接到同一个路由。

在 PC 上使用网络调试工具，创建一个 UDP 传输。例如 PC 的 IP 地址为 192.168.3.102，端口为 8080。

### 5. 使能多连接。

AT

```
1 AT+CIPMUX=1
```

Plain Text

```
1 OK
```

### 6. 创建 UDP 传输。分配网络连接 ID 为 4，远程 IP 地址为 192.168.3.102，远端端口为 8080，本地端口为 1112，模式为 0。

**重要**

AT

```
1 AT+CIPSTART 命令的参数 mode 决定了 UDP 通信的远端 IP 地址和端口是否固定。若参数为 0，
   则代表系统会分配一个特定网络连接 ID，以确保通信过程中远端的 IP 地址和端口不会被改变，且
   数据发送端和数据接收端不会被其它设备代替。
```

AT

```
1 AT+CIPSTART=4,"UDP", "192.168.3.102",8080,1112,0
```

Plain Text

```
1 4,CONNECT
2 OK
```

**说明：**"192.168.3.102" 和 8080 为 UDP 传输的远端 IP 地址和远端端口，也就是 PC 建立的 UDP 配置。1112 为 XY-MWC320A 设备的 UDP 本地端口，可自行设置，如不设置则为随机值。0 表示 UDP 远端 IP 地址和端口是固定的，不能更改。比如有另外一台 PC 创建了 UDP 端并且向 XY-MWC320A 设备端口 1112 发送数据，XY-MWC320A 设备仍然会接收来自 UDP 端口 1112 的数据，如果使用 AT 命令 AT+CIPSEND=4,X，那么数据仍然只会发送到第一台 PC 端。但是如果这个参数未设置为 0，那么数据将会被发送到新的 PC 端。

## 7. 发送 7 字节数据到网络连接 ID 为 4 的链路上。

AT

```
1 AT+CIPSEND=4,7
```

Plain Text

```
1 OK
2 >
```

输入 7 字节数据，例如输入数据是 abcdefg，之后 AT 将会输出以下信息。

Plain Text

```
1 Recv 7 bytes
2 SEND OK
```

**说明：**若输入的字节数目超过 AT+CIPSEND 命令设定的长度 (n)，则系统会响应 busy p...，并发送数据的前 n 个字节，发送完成后响应 SEND OK。

## 8. 从网络连接 ID 为 4 的链路上接收 4 字节数据。

假设 PC 发送 4 字节的数据（数据为 test），则系统会提示：

Plain Text

```
1 +IPD,4,4:test
```

## 9. 关闭网络连接 ID 为 4 的 UDP 连接。

AT

```
1 AT+CIPCLOSE=4
```

Plain Text

```
1 4,CLOSED
2 OK
```

### 2.3.4 远端 IP 地址和端口可变的 UDP 通信

## 1. 设置 Wi-Fi 模式为 station。

AT

1 AT+CWMODE=1

Plain Text

1 OK

## 2. 连接到路由器。

AT

1 AT+CWJAP="XY-Router","1234567890"

Plain Text

1 WIFI CONNECTED  
2 WIFI GOT IP  
3 OK

**说明：**实际输入的 SSID 和密码可能与上述命令中的不同，请使用所用路由器的 SSID 和密码。

## 3. 查询 XY-MWC320A 设备 IP 地址。

AT

1 AT+CIPSTA?

Plain Text

1 +CIPSTA:ip:"192.168.3.112"  
2 +CIPSTA:gateway:"192.168.3.1"  
3 +CIPSTA:netmask:"255.255.255.0"  
4 OK

**说明：**查询结果可能与上述响应中的不同。

## 4. PC 与 XY-MWC320A 设备连接到同一个路由。

在 PC 上使用网络调试工具，创建一个 UDP 传输。例如 IP 地址为 192.168.3.102，端口为 8080。

## 5. 使能单连接。

AT

1 AT+CIPMUX=0

Plain Text

1 OK

## 6. 创建 UDP 传输。远程 IP 地址为 192.168.3.102，远端端口为 8080，本地端口为 1112，模式为 2。

AT

1 AT+CIPSTART="UDP","192.168.3.102",8080,1112,2

Plain Text

- 1 CONNECT
- 2 OK

**说明：**"192.168.3.102" 和 8080 为 UDP 传输的远端 IP 地址和远端端口，也就是 PC 建立的 UDP 配置。1112 为 XY-MWC320A 设备的 UDP 本地端口，可自行设置，如不设置则为随机值。2 表示当前 UDP 传输建立后，UDP 传输远端信息仍然会更改；UDP 传输的远端信息会自动更改为最近一次与 XY-MWC320A 设备 UDP 通信的远端 IP 地址和端口。

## 7. 发送 4 字节数据。

AT

- 1 AT+CIPSEND=4

Plain Text

- 1 OK
- 2 >

输入 4 字节数据，例如输入数据是 test，之后 AT 将会输出以下信息。

Plain Text

- 1 Recv 4 bytes
- 2 SEND OK

**说明：**若输入的字节数目超过 AT+CIPSEND 命令设定的长度 (n)，则系统会响应 busy p...，并发送数据的前 n 个字节，发送完成后响应 SEND OK。

## 8. 发送 UDP 包给其它 UDP 远端。例如发送 4 字节数据，远端主机的 IP 地址为 192.168.3.103，远端端口为 1000。

若需要发 UDP 包给其它 UDP 远端，只需指定对方 IP 地址和端口即可。

AT

- 1 AT+CIPSEND=4,"192.168.3.103",1000

Plain Text

- 1 OK
- 2 >

输入 4 字节数据，例如输入数据是 test，之后 AT 将会输出以下信息。

Plain Text

- 1 Recv 4 bytes
- 2 SEND OK

## 9. 接收 4 字节数据。

假设 PC 发送 4 字节的数据（数据为 test），则系统会提示：

Plain Text

```
1 +IPD,4:test
```

## 10. 关闭 UDP 连接。

AT

```
1 AT+CIPCLOSE
```

Plain Text

```
1 CLOSED
2 OK
```

### 2.3.5 XY-MWC320A 设备作为 SSL 客户端建立单连接

#### 1. 设置 Wi-Fi 模式为 station。

AT

```
1 AT+CWMODE=1
```

Plain Text

```
1 OK
```

#### 2. 连接到路由器。

AT

```
1 AT+CWJAP="XY-Router","1234567890"
```

Plain Text

```
1 WIFI CONNECTED
2 WIFI GOT IP
3 OK
```

**说明：**实际输入的 SSID 和密码可能与上述命令中的不同，请使用所用路由器的 SSID 和密码。

#### 3. 查询 XY-MWC320A 设备 IP 地址。

AT

```
1 AT+CIPSTA?
```

Plain Text

```
1 +CIPSTA:ip:"192.168.3.112"
2 +CIPSTA:gateway:"192.168.3.1"
3 +CIPSTA:netmask:"255.255.255.0"
4 OK
```

**说明：**查询结果可能与上述响应中的不同。

#### 4. PC 与 XY-MWC320A 设备连接同一个路由。

5. 在 PC 上使用 OpenSSL 命令，创建一个 SSL 服务器。例如 SSL 服务器的 IP 地址为 192.168.3.102，端口为 8070。

```
openssl s_server -cert /home/XY-MWC320A/components/customized_partitions/raw_data/server_cert/server_cert.crt -key /home/XY-MWC320A/components/customized_partitions/raw_data/server_key/server.key -port 8070
```

Plain Text

```
1 ACCEPT
```

6. XY-MWC320A 设备作为客户端通过 SSL 连接到 SSL 服务器，服务器 IP 地址为 192.168.3.102，端口为 8070。

AT

```
1 AT+CIPSTART="SSL","192.168.3.102",8070
```

Plain Text

```
1 CONNECT
2 OK
```

7. 发送 4 字节数据。

AT

```
1 AT+CIPSEND=4
```

Plain Text

```
1 OK
2 >
```

输入 4 字节数据，例如输入数据是 test，之后 AT 将会输出以下信息。

Plain Text

```
1 Recv 4 bytes
2 SEND OK
```

**说明：**若输入的字节数目超过 AT+CIPSEND 命令设定的长度 (n)，则系统会响应 busy p...，并发送数据的前 n 个字节，发送完成后响应 SEND OK。

8. 接收 4 字节数据。

假设 TCP 服务器发送 4 字节的数据（数据为 test），则系统会提示：

Plain Text

```
1 +IPD,4:test
```

### 2.3.6 XY-MWC320A 设备作为 SSL 服务器建立多连接

当 XY-MWC320A 设备作为 SSL 服务器时，必须通过 AT+CIPMUX=1 命令使能多连接，因为可能有多  
个客户端连接到 XY-MWC320A 设备。

以下是 XY-MWC320A 设备作为 softAP 建立 SSL 服务器的示例；如果是 XY-MWC320A 设备作为  
station，可在连接路由器后，参照本示例中的建立连接 SSL 服务器的相关步骤。

### 1. 设置 Wi-Fi 模式为 softAP。

```
AT
1 AT+CWMODE=2
```

```
Plain Text
1 OK
```

### 2. 使能多连接。

```
AT
1 AT+CIPMUX=1
```

```
Plain Text
1 OK
```

### 3. 配置 XY-MWC320A softAP。

```
AT
1 AT+CWSAP="XY_softAP","1234567890",5,3
```

```
Plain Text
1 OK
```

### 4. 查询 softAP 信息。

```
AT
1 AT+CIPAP?
```

```
Plain Text
1 AT+CIPAP?
2 +CIPAP:ip:"192.168.4.1"
3 +CIPAP:gateway:"192.168.4.1"
4 +CIPAP:netmask:"255.255.255.0"
5 OK
```

**说明：** 查询到的地址可能与上述响应中的不同。

### 5. 建立 SSL 服务器，端口为 8070。

```
AT
1 AT+CIPSERVER=1,8070,"SSL"
```

Plain Text

1 OK

## 6. PC 连接到 XY-MWC320A 设备的 softAP。

## 7. 在 PC 上使用 OpenSSL 命令，创建一个 SSL 客户端，连接到 XY-MWC320A 设备创建的 SSL 服务器。

```
openssl s_client -host 192.168.4.1 -port 8070
```

XY-MWC320A 设备上的响应：

Plain Text

1 CONNECT

## 8. 发送 4 字节数据到网络连接 ID 为 0 的链路上。

AT

1 AT+CIPSEND=0,4

Plain Text

1 OK

2 >

输入 4 字节数据，例如输入数据是 test，之后 AT 将会输出以下信息。

Plain Text

1 Recv 4 bytes

2 SEND OK

**说明：**若输入的字节数目超过 AT+CIPSEND 命令设定的长度 (n)，则系统会响应 busy p...，并发送数据的前 n 个字节，发送完成后响应 SEND OK。

## 9. 从网络连接 ID 为 0 的链路上接收 4 字节数据。

假设 SSL 服务器发送 4 字节的数据（数据为 test），则系统会提示：

Plain Text

1 +IPD,0,4:test

## 10. 关闭 SSL 连接。

AT

1 AT+CIPCLOSE=0

Plain Text

1 0,CLOSED

2 OK

### 2.3.7 XY-MWC320A 设备作为 SSL 客户端建立双向认证单连接

本示例中使用的证书是 XY-MWC320A 中默认的证书，也可以使用自己的证书：

- 要使用自己的 SSL 客户端证书，可以在运行时使用 AT+SYSMFG 命令更新证书（请参考 AT+SYSMFG 命令示例中的 PKI 配置），也可以通过重新编译固件替换默认证书（请参考 如何更新 PKI 配置）。
- 要使用自己的 SSL 服务器证书，请用自己的证书路径替换下面的 SSL 服务器证书。

#### 1. 设置 Wi-Fi 模式为 station。

AT

```
1 AT+CWMODE=1
```

Plain Text

```
1 OK
```

#### 2. 连接到路由器。

AT

```
1 AT+CWJAP="XY-Router","1234567890"
```

Plain Text

```
1 WIFI CONNECTED
2 WIFI GOT IP
3 OK
```

**说明：**实际输入的 SSID 和密码可能与上述命令中的不同，请使用所用路由器的 SSID 和密码。

#### 3. 设置 SNTP 服务器。

AT

```
1 AT+CIPSNTPCFG=1,8,"cn.ntp.org.cn","ntp.sjtu.edu.cn"
```

Plain Text

```
1 OK
```

**说明：**可根据所在国家的时区设置 SNTP 服务器。

#### 4. 查询 SNTP 时间。

AT

```
1 AT+CIPSNTPTIME?
```

Plain Text

```
1 +CIPSNTPTIME:Mon Oct 18 20:12:27 2021
2 OK
```

**说明：**可以查询 SNTP 时间与实时时间是否相符来判断所设置的 SNTP 服务器是否生效。

## 5. 查询 XY-MWC320A 设备 IP 地址。

AT

1 AT+CIPSTA?

Plain Text

```
1 +CIPSTA:ip:"192.168.3.112"
2 +CIPSTA:gateway:"192.168.3.1"
3 +CIPSTA:netmask:"255.255.255.0"
4 OK
```

**说明：**查询结果可能与上述响应中的不同。

## 6. PC 与 XY-MWC320A 设备连接同一个路由。

### 7. 在 PC 上使用 OpenSSL 命令，创建一个 SSL 服务器。例如 SSL 服务器的 IP 地址为 192.168.3.102，端口为 8070。

```
openssl s_server -CAfile /home/XY-MWC320A/components/customized_partitions/raw_data/server_ca/server_ca.crt -cert /home/XY-MWC320A/components/customized_partitions/raw_data/server_cert/server_cert.crt -key /home/XY-MWC320A/components/customized_partitions/raw_data/server_key/server.key -port 8070 -verify_return_error -verify_depth 1 -Verify 1
```

XY-MWC320A 设备上的响应：

Plain Text

1 ACCEPT

**说明：**命令中的证书路径可以根据的证书位置进行调整。

## 8. XY-MWC320A 设备设置 SSL 客户端双向认证配置。

AT

1 AT+CIPSSLCCNF=3,0,0

Plain Text

1 OK

### 9. XY-MWC320A 设备作为客户端通过 SSL 连接到 SSL 服务器，服务器 IP 地址为 192.168.3.102，端口为 8070。

AT

1 AT+CIPSTART="SSL","192.168.3.102",8070

Plain Text

```
1 CONNECT
2 OK
```

## 10. 发送 4 字节数据。

AT

```
1 AT+CIPSEND=4
```

Plain Text

```
1 OK
2 >
```

输入 4 字节数据，例如输入数据是 test，之后 AT 将会输出以下信息。

Plain Text

```
1 Recv 4 bytes
2 SEND OK
```

**说明：**若输入的字节数目超过 AT+CIPSEND 命令设定的长度 (n)，则系统会响应 busy p...，并发送数据的前 n 个字节，发送完成后响应 SEND OK。

## 11. 接收 4 字节数据。

假设 TCP 服务器发送 4 字节的数据（数据为 test），则系统会提示：

Plain Text

```
1 +IPD,4:test
```

### 2.3.8 XY-MWC320A 设备作为 SSL 服务器建立双向认证多连接

当 XY-MWC320A 设备作为 SSL 服务器时，必须通过 AT+CIPMUX=1 命令使能多连接，因为可能有多客户端连接到 XY-MWC320A 设备。

以下是 XY-MWC320A 设备作为 station 建立 SSL 服务器的示例；如果是 XY-MWC320A 设备作为 softAP，可参考 XY-MWC320A 设备作为 SSL 服务器建立多连接 示例。

#### 1. 设置 Wi-Fi 模式为 station。

AT

```
1 AT+CWMODE=1
```

Plain Text

```
1 OK
```

#### 2. 连接到路由器。

AT

1 AT+CWJAP="XY-Router","1234567890"

Plain Text

1 WIFI CONNECTED  
2 WIFI GOT IP  
3 OK

**说明：**实际输入的 SSID 和密码可能与上述命令中的不同，请使用所用路由器的 SSID 和密码。

### 3. 查询 XY-MWC320A 设备 IP 地址。

AT

1 AT+CIPSTA?

Plain Text

1 +CIPSTA:ip:"192.168.3.112"  
2 +CIPSTA:gateway:"192.168.3.1"  
3 +CIPSTA:netmask:"255.255.255.0"  
4 OK

**说明：**查询结果可能与上述响应中的不同。

### 4. 使能多连接。

AT

1 AT+CIPMUX=1

Plain Text

1 OK

### 5. 建立 SSL 服务器，端口为 8070。

AT

1 AT+CIPSERVER=1,8070,"SSL",1

Plain Text

1 OK

### 6. PC 与 XY-MWC320A 设备连接同一个路由。

### 7. 在 PC 上使用 OpenSSL 命令，创建一个 SSL 客户端，连接到 XY-MWC320A 设备创建的 SSL 服务器。

```
openssl s_client -CAfile /home/XY-MWC320A/components/customized_partitions/raw_data/client_ca/client_ca_00.crt -cert /home/XY-
```

```
MWC320A/components/customized_partitions/raw_data/client_cert/client_cert_00.crt -key  
/home/XY-  
MWC320A/components/customized_partitions/raw_data/client_key/client_key_00.key -host  
192.168.3.112 -port 8070
```

XY-MWC320A 设备上的响应:

```
0,CONNECT
```

#### 8. 发送 4 字节数据到网络连接 ID 为 0 的链路上。

```
AT  
1 AT+CIPSEND=0,4
```

Plain Text

```
1 OK  
2 >
```

输入 4 字节数据，例如输入数据是 test，之后 AT 将会输出以下信息。

```
Plain Text  
1 Recv 4 bytes  
2 SEND OK
```

**说明：**若输入的字节数目超过 AT+CIPSEND 命令设定的长度 (n)，则系统会响应 busy p...，并发送数据的前 n 个字节，发送完成后响应 SEND OK。

#### 9. 从网络连接 ID 为 0 的链路上接收 4 字节数据。

假设 SSL 服务器发送 4 字节的数据（数据为 test），则系统会提示：

```
Plain Text  
1 +IPD,0,4:test
```

#### 10. 关闭 SSL 连接。

```
AT  
1 AT+CIPCLOSE=0
```

Plain Text

```
1 0,CLOSED  
2 OK
```

#### 11. 关闭 SSL 服务端。

```
AT  
1 AT+CIPSERVER=0
```

Plain Text

1 OK

### 2.3.9 XY-MWC320A 设备作为 TCP 客户端，建立单连接，实现 Network 透传

#### 1. 设置 Wi-Fi 模式为 station。

AT

1 AT+CWMODE=1

Plain Text

1 OK

#### 2. 连接到路由器。

AT

1 AT+CWJAP="XY-Router","1234567890"

Plain Text

1 WIFI CONNECTED  
2 WIFI GOT IP  
3 OK

**说明：**实际输入的 SSID 和密码可能与上述命令中的不同，请使用所用路由器的 SSID 和密码。

#### 3. 查询 XY-MWC320A 设备 IP 地址。

AT

1 AT+CIPSTA?

Plain Text

1 +CIPSTA:ip:"192.168.3.112"  
2 +CIPSTA:gateway:"192.168.3.1"  
3 +CIPSTA:netmask:"255.255.255.0"  
4 OK

**说明：**查询结果可能与上述响应中的不同。

#### 4. PC 与 XY-MWC320A 设备连接到同一个路由。

在 PC 上使用网络调试工具，创建一个 TCP 服务器。例如 IP 地址为 192.168.3.102，端口为 8080。

#### 5. XY-MWC320A 设备作为客户端通过 TCP 连接到 TCP 服务器，服务器 IP 地址为 192.168.3.102，端口为 8080。

AT

1 AT+CIPSTART="TCP","192.168.3.102",8080

Plain Text

- 1 CONNECT
- 2 OK

## 6. 进入 Network 透传接收模式。

AT

- 1 AT+CIPMODE=1

Plain Text

- 1 OK

## 7. 进入 Network 透传模式 并发送数据。

AT

- 1 AT+CIPSEND

Plain Text

- 1 OK
- 2 >

## 8. 停止发送数据

在透传发送数据过程中，若识别到单独的一包数据 +++，则系统会退出透传发送。此时请至少等待 1 秒，再发下一条 AT 命令。请注意，如果直接用键盘打字输入 +++，有可能因时间太慢而不能被识别为连续的三个 +。更多介绍请参考 [仅适用透传模式] +++。

### 重要

使用 +++ 可退出 透传模式，回到 透传接收模式，此时 TCP 连接仍然有效。也可以使用 AT+CIPSEND 命令恢复透传。

## 9. 退出 Network 透传接收模式。

AT

- 1 AT+CIPMODE=0

Plain Text

- 1 OK

## 10. 关闭 TCP 连接。

AT

- 1 AT+CIPCLOSE

Plain Text

- 1 CLOSED
- 2 OK

### 2.3.10 XY-MWC320A 设备作为 TCP 服务器，实现 Network 透传

#### 1. 设置 Wi-Fi 模式为 station。

AT

1 AT+CWMODE=1

Plain Text

1 OK

#### 2. 连接到路由器。

AT

1 AT+CWJAP="XY-Router", "1234567890"

Plain Text

1 WIFI CONNECTED

2 WIFI GOT IP

3 OK

**说明：**实际输入的 SSID 和密码可能与上述命令中的不同，请使用所用路由器的 SSID 和密码。

#### 3. 设置多连接模式。

AT

1 AT+CIPMUX=1

Plain Text

1 OK

**说明：**TCP 服务器必须在多连接模式下才能开启。

#### 4. 设置 TCP 服务器最大连接数为 1。

AT

1 AT+CIPSERVERMAXCONN=1

Plain Text

1 OK

**说明：**透传模式是点对点的，因此 TCP 服务器的最大连接数只能是 1。

#### 5. 开启 TCP 服务器。

AT

1 AT+CIPSERVER=1,8080

Plain Text

1 OK

**说明：**设置 TCP 服务器端口为 8080，也可以设置为其它端口。

## 6. 查询 XY-MWC320A 设备 IP 地址。

AT

1 AT+CIPSTA?

Plain Text

```
1 +CIPSTA:ip:"192.168.3.112"
2 +CIPSTA:gateway:"192.168.3.1"
3 +CIPSTA:netmask:"255.255.255.0"
4 OK
```

**说明：**查询结果可能与上述响应中的不同。

## 7. PC 连接到 XY-MWC320A TCP 服务器。

PC 与 XY-MWC320A 设备连接到同一个路由。

在 PC 上使用网络调试工具，创建一个 TCP 客户端。连接到 XY-MWC320A 的 TCP 服务器。地址为 192.168.3.112，端口为 8080。

AT

1 AT 响应：

0,CONNECT

## 8. 进入 Network 透传接收模式。

AT

1 AT+CIPMODE=1

Plain Text

1 OK

## 9. 进入 Network 透传模式 并发送数据。

AT

1 AT+CIPSEND

Plain Text

```
1 OK
2 >
```

## 10. 停止发送数据

在透传发送数据过程中，若识别到单独的一包数据 +++，则系统会退出透传发送。此时请至少等待 1 秒，再发下一条 AT 命令。请注意，如果直接用键盘打字输入 +++，有可能因时间太慢而不能被识别为连续的两个 +。更多介绍请参考 [仅适用透传模式] +++。

## 重要

使用 +++ 可退出 透传模式，回到 透传接收模式，此时 TCP 连接仍然有效。也可以使用 AT+CIPSEND 命令恢复透传。

### 11. 退出 Network 透传接收模式。

```
AT
1 AT+CIPMODE=0
```

```
Plain Text
1 OK
```

### 12. 关闭 TCP 连接。

```
AT
1 AT+CIPCLOSE
```

```
Plain Text
1 CLOSED
2 OK
```

## 2.3.11 XY-MWC320A 设备作为 softAP 在 UDP 传输中实现 Network 透传

### 1. 设置 Wi-Fi 模式为 softAP。

```
AT
1 AT+CWMODE=2
```

```
Plain Text
1 OK
```

### 2. 设置 softAP。

```
AT
1 AT+CWSAP="XY_softAP", "1234567890", 5, 3
```

```
Plain Text
1 OK
```

### 3. PC 连接到 XY-MWC320A 设备的 softAP。

### 4. 创建一个 UDP 端点。

在 PC 上使用网络调试助手，创建一个 UDP 传输。例如 PC 端 IP 地址为 192.168.4.2，端口为 8080。

5. XY-MWC320A 与 PC 对应端口建立固定对端 IP 地址和端口的 UDP 传输。远程 IP 地址为 192.168.4.2，远端端口为 8080，本地端口为 2233，模式为 0。

AT

1 AT+CIPSTART="UDP","192.168.4.2",8080,2233,0

Plain Text

1 CONNECT

2 OK

6. 进入 Network 透传接收模式。

AT

1 AT+CIPMODE=1

Plain Text

1 OK

7. 进入 Network 透传模式 并发送数据。

AT

1 AT+CIPSEND

Plain Text

1 OK

2 >

8. 停止发送数据

在透传发送数据过程中，若识别到单独的一包数据 +++，则系统会退出透传发送。此时请至少等待 1 秒，再发下一条 AT 命令。请注意，如果直接用键盘打字输入 +++，有可能因时间太慢而不能被识别为连续的三个 +。更多介绍请参考 [仅适用透传模式] +++。

**重要**

使用 +++ 可退出 透传模式，回到 透传接收模式，此时 TCP 连接仍然有效。也可以使用 AT+CIPSEND 命令恢复透传。

9. 退出 Network 透传接收模式。

AT

1 AT+CIPMODE=0

Plain Text

1 OK

10. 关闭 TCP 连接。

AT

1 AT+CIPCLOSE

Plain Text

1 CLOSED

2 OK

### 2.3.12 XY-MWC320A 设备获取被动接收模式下的套接字数据

预计设备将接收大量网络数据并且 MCU 端来不及处理时，可以参考该示例，使用被动接收数据模式。

#### 1. 设置 Wi-Fi 模式为 station。

AT

1 AT+CWMODE=1

Plain Text

1 OK

#### 2. 连接到路由器。

AT

1 AT+CWLAP="XY-Router", "1234567890"

Plain Text

1 WIFI CONNECTED

2 WIFI GOT IP

3 OK

**说明：**实际输入的 SSID 和密码可能与上述命令中的不同，请使用所用路由器的 SSID 和密码。

#### 3. 查询 XY-MWC320A 设备 IP 地址。

AT

1 AT+CIPSTA?

Plain Text

1 +CIPSTA:ip:"192.168.3.112"

2 +CIPSTA:gateway:"192.168.3.1"

3 +CIPSTA:netmask:"255.255.255.0"

4 OK

**说明：**查询结果可能与上述响应中的不同。

#### 4. PC 与 XY-MWC320A 设备连接同一个路由。

在 PC 上使用网络调试工具，创建一个 TCP 服务器。例如 TCP 服务器的 IP 地址为 192.168.3.102，端口为 8080。

5. XY-MWC320A 设备作为客户端通过 TCP 连接到 TCP 服务器，服务器 IP 地址为 192.168.3.102，端口为 8080。

AT

1 AT+CIPSTART="TCP","192.168.3.102",8080

Plain Text

1 CONNECT

2 OK

6. XY-MWC320A 设备设置套接字接收模式为被动模式。

AT

1 AT+CIPRECVMODE=1

Plain Text

1 OK

7. TCP 服务器发送 4 字节的数据（数据为 test）。

**说明：**此时会回复 +IPD,4，如果后续再接收到服务器数据，是否回复 +IPD,, 请阅读 AT+CIPRECVMODE 说明部分。

8. XY-MWC320A 设备查询被动接收模式下套接字数据的长度。

AT

1 AT+CIPRCVLEN?

Plain Text

1 +CIPRCVLEN:4

2 OK

9. XY-MWC320A 设备获取被动接收模式下的套接字数据。

AT

1 AT+CIPRCVDATA=4

Plain Text

1 +CIPRCVDATA:4,test

2 OK

### 2.3.13 XY-MWC320A 设备开启 mDNS 服务，PC 通过域名连接到设备的 TCP 服务器

1. 设置 Wi-Fi 模式为 station。

AT

1 AT+CWMODE=1

Plain Text

1 OK

## 2. 连接到路由器。

AT

1 AT+CWJAP="XY-Router","1234567890"

Plain Text

1 WIFI CONNECTED  
2 WIFI GOT IP  
3 OK

**说明：**实际输入的 SSID 和密码可能与上述命令中的不同，请使用所用路由器的 SSID 和密码。

## 3. PC 与 XY-MWC320A 设备连接同一个路由。

只有在同一个局域网中，PC 才能发现 XY-MWC320A 设备。

## 4. 在 PC 上使用 mDNS 工具开启服务发现。例如 Linux 上使用 avahi-browse（macOS 或 Windows 上使用 Bonjour）。

`sudo avahi-browse -a -r`

## 5. XY-MWC320A 设备开启 mDNS 服务。

AT

1 AT+MDNS=1,"XY","\_printer",35,"my\_instance","\_tcp",2,"product","my\_printer","firmware\_version","AT-V3.4.1.0"

Plain Text

1 OK

**说明：**此命令开启 mDNS 服务，表明了设备实例名称为 my\_instance，服务类型为 \_printer，端口为 35，产品为 my\_printer，固件版本为 AT-V3.4.1.0。

## 6. （可选步骤）PC 端发现 XY-MWC320A 设备。

PC 端的 avahi-browse 工具会提示：

...

+ enx000ec6dd4ebf IPv4 my\_instance UNIX Printer local

```
AT
1 = enx000ec6dd4ebf IPv4 my_instance          UNIX Printe
r          local
2  hostname = [XY.local]
3  address = [192.168.200.90]
4  port = [35]
5  txt = ["product=my_printer" "firmware_version=AT-V3.4.1.0"]
```

**说明：**此步骤不是必须的，只是为了验证 XY-MWC320A 设备的 mDNS 功能是否正常。

## 7. XY-MWC320A 设备使能多连接。

```
AT
1 AT+CIPMUX=1
```

```
Plain Text
1 OK
```

## 8. XY-MWC320A 设备作为 TCP 服务器，端口为 35。

```
AT
1 AT+CIPSERVER=1,35
```

```
Plain Text
1 OK
```

## 9. 在 PC 上使用 TCP 工具（例如，Linux 或 macOS 上使用 nc，Windows 上使用 ncat），通过域名连接到 XY-MWC320A 设备的 TCP 服务器。

```
nc XY.local 35
```

XY-MWC320A 设备响应：

```
0,CONNECT
```

**说明：**连接建立成功后，PC 和 XY-MWC320A 设备之间立即可以进行数据传输。

## 10. XY-MWC320A 设备关闭 mDNS 功能。

```
AT
1 AT+MDNS=0
```

```
Plain Text
1 OK
```

**说明：**关闭 mDNS 功能能一定程度上减少设备的功耗。

## 2.4 Bluetooth LE AT 示例

本文档主要介绍 Bluetooth® Low Energy AT 命令集 的使用方法，并提供在 XY-MWC320A 设备上运行这些命令的详细示例。

## 2.4.1 简介

当前，XY-MWC320A AT 固件支持 蓝牙核心规范 5.0 版本。

### 2.4.1.1 Bluetooth LE 协议架构

Bluetooth LE 协议栈从下至上分为几个层级：Physical Layer (PHY)、Link Layer (LL)、Host Controller Interface (HCI)、Logical Link Control and Adaptation Protocol Layer (L2CAP)、Attribute Protocol (ATT)、Security Manager Protocol (SMP)、Generic Attribute Profile (GATT)、Generic Access Profile (GAP)。

- PHY: PHY 层主要负责在物理信道上发送和接收信息包。Bluetooth LE 使用 40 个射频信道。频率范围：2402 MHz 到 2480 MHz。
- LL: LL 层主要负责创建、修改和释放逻辑链路（以及，如果需要，它们相关的逻辑传输），以及与设备之间的物理链路相关的参数的更新。它控制链路层状态机处于 准备、广播、监听/扫描、发起连接、已连接 五种状态之一。
- HCI: HCI 层向主机和控制器提供一个标准化的接口。该层可以由软件 API 实现或者使用硬件接口 UART、SPI、USB 来控制。
- L2CAP: L2CAP 层负责对主机和协议栈之间交换的数据进行协议复用能力、分段和重组操作。
- ATT: ATT 层实现了属性服务器和属性客户端之间的点对点协议。ATT 客户端向 ATT 服务端发送命令、请求和确认。ATT 服务端向客户端发送响应、通知和指示。
- SMP: SMP 层用于生成加密密钥和身份密钥。SMP 还管理加密密钥和身份密钥的存储，并负责生成随机地址并将随机地址解析为已知设备身份。
- GATT: GATT 层表示属性服务器和可选的属性客户端的功能。该配置文件描述了属性服务器中使用的服务、特征和属性的层次结构。该层提供用于发现、读取、写入和指示服务特性和属性的接口。
- GAP: GAP 层代表所有蓝牙设备通用的基本功能，例如传输、协议和应用程序配置文件使用的模式和访问程序。GAP 服务包括设备发现、连接模式、安全、身份验证、关联模型和服务发现。

### 2.4.1.2 Bluetooth LE 角色划分

在 Bluetooth LE 协议栈中不同的层级有不同的角色划分。这些角色划分互不影响。

- LL: 设备可以划分为 主机 和 从机，从机广播，主机可以发起连接。
- GAP: 定义了 4 种特定角色：广播者、观察者、外围设备 和 中心设备。
- GATT: 设备可以分为 服务端 和 客户端。

**重要**

- 本文档中描述的 Bluetooth LE 服务端 和 Bluetooth LE 客户端 都是 GATT 层角色。
- 当前，XY-MWC320A 支持 Bluetooth LE 服务端 和 Bluetooth LE 客户端 同时存在。
- 不论 XY-MWC320A 初始化为 Bluetooth LE 服务端还是 Bluetooth LE 客户端，同时连接的最大设备数量为 3。

GATT 其实是一种属性传输协议，简单的讲可以认为是一种属性传输的应用层协议。这个属性的结构非常简单。它由 服务 组成，每个服务由不同数量的 特征 组成，每个 特征 又由很多其它的元素组成。

GATT 服务端 和 GATT 客户端 这两种角色存在于 Bluetooth LE 连接建立之后。GATT 服务端存储通过属性协议传输的数据，并接受来自 GATT 客户端的属性协议请求、命令和确认。简而言之，提供数据的一端称为 GATT 服务端，访问数据的一端称为 GATT 客户端。

### 2.4.2 Bluetooth LE 客户端读写服务特征值

以下示例同时使用两块 XY-MWC320A 开发板，其中一块作为 Bluetooth LE 服务端（只作为 Bluetooth LE 服务端角色），另一块作为 Bluetooth LE 客户端（只作为 Bluetooth LE 客户端角色）。这个例子展示了应如何使用 AT 命令建立 Bluetooth LE 连接，完成数据通信。

#### 重要

在以下步骤中以 XY-MWC320A Bluetooth LE 服务端 开头的操作只需要在 XY-MWC320A Bluetooth LE 服务端执行即可，以 XY-MWC320A Bluetooth LE 客户端 开头的操作只需要在 XY-MWC320A Bluetooth LE 客户端执行即可。

#### 1. 初始化 Bluetooth LE 功能。

XY-MWC320A Bluetooth LE 服务端：

```
AT
1 AT+BLEINIT=2
```

```
Plain Text
1 OK
```

XY-MWC320A Bluetooth LE 客户端：

```
AT
1 AT+BLEINIT=1
```

```
Plain Text
1 OK
```

#### 2. XY-MWC320A Bluetooth LE 服务端获取其 MAC 地址。

```
AT
1 AT+BLEADDR?
```

Plain Text

```
1 +BLEADDR:"24:0a:c4:d6:e4:46"  
2 OK
```

**说明：**查询到的地址可能与上述响应中的不同，请记住该地址，后续步骤中会用到。

### 3. XY-MWC320A Bluetooth LE 服务端创建服务。

AT

```
1 AT+BLEGATTSSRVCRE
```

Plain Text

```
1 OK
```

### 4. XY-MWC320A Bluetooth LE 服务端开启服务。

AT

```
1 AT+BLEGATTSSRVSTART
```

Plain Text

```
1 OK
```

### 5. XY-MWC320A Bluetooth LE 服务端发现服务特征。

AT

```
1 AT+BLEGATTSCCHAR?
```

Plain Text

```
1 +BLEGATTSCCHAR:"char",1,1,0xC300,0x02
2 +BLEGATTSCCHAR:"desc",1,1,1,0x2901
3 +BLEGATTSCCHAR:"char",1,2,0xC301,0x02
4 +BLEGATTSCCHAR:"desc",1,2,1,0x2901
5 +BLEGATTSCCHAR:"char",1,3,0xC302,0x08
6 +BLEGATTSCCHAR:"desc",1,3,1,0x2901
7 +BLEGATTSCCHAR:"char",1,4,0xC303,0x04
8 +BLEGATTSCCHAR:"desc",1,4,1,0x2901
9 +BLEGATTSCCHAR:"char",1,5,0xC304,0x08
10 +BLEGATTSCCHAR:"char",1,6,0xC305,0x10
11 +BLEGATTSCCHAR:"desc",1,6,1,0x2902
12 +BLEGATTSCCHAR:"char",1,7,0xC306,0x20
13 +BLEGATTSCCHAR:"desc",1,7,1,0x2902
14 +BLEGATTSCCHAR:"char",1,8,0xC307,0x02
15 +BLEGATTSCCHAR:"desc",1,8,1,0x2901
16 +BLEGATTSCCHAR:"char",2,1,0xC400,0x02
17 +BLEGATTSCCHAR:"desc",2,1,1,0x2901
18 +BLEGATTSCCHAR:"char",2,2,0xC401,0x02
19 +BLEGATTSCCHAR:"desc",2,2,1,0x2901
20 OK
```

6. XY-MWC320A Bluetooth LE 服务端开始广播，之后 XY-MWC320A Bluetooth LE 客户端开始扫描并且持续 3 秒钟。

XY-MWC320A Bluetooth LE 服务端：

AT

```
1 AT+BLEADVSTART
```

Plain Text

```
1 OK
```

XY-MWC320A Bluetooth LE 客户端：

AT

```
1 AT+BLESCAN=1,3
```

Plain Text

```
1 OK
2 +BLESCAN:"5b:3b:6c:51:90:49",-87,02011a020a0c0aff4c001005071c3024dc,,1
3 +BLESCAN:"c4:5b:be:93:ec:66",-84,0201060303111809095647543147572d58020a03,,0
4 +BLESCAN:"24:0a:c4:d6:e4:46",-29,,,0
```

**说明：**的扫描结果可能与上述响应中的不同。

## 7. 建立 Bluetooth LE 连接。

## XY-MWC320A Bluetooth LE 客户端:

AT

```
1 AT+BLECONN=0,"24:0a:c4:d6:e4:46"
```

Plain Text

```
1 +BLECONN:0,"24:0a:c4:d6:e4:46"
2 OK
```

**说明:** 输入上述命令时, 请使用的 XY-MWC320A Bluetooth LE 服务端地址。如果 Bluetooth LE 连接成功, 则会提示 +BLECONN:0,"24:0a:c4:d6:e4:46。如果 Bluetooth LE 连接失败, 则会提示 +BLECONN:0,-1。

## 8. XY-MWC320A Bluetooth LE 客户端发现服务。

AT

```
1 AT+BLEGATTCPRIMSRV=0
```

Plain Text

```
1 +BLEGATTCPRIMSRV:0,1,0x1801,1
2 +BLEGATTCPRIMSRV:0,2,0x1800,1
3 +BLEGATTCPRIMSRV:0,3,0xA002,1
4 +BLEGATTCPRIMSRV:0,4,0xA003,1
5 OK
```

**说明:** XY-MWC320A Bluetooth LE 客户端查询服务的结果, 比 XY-MWC320A Bluetooth LE 服务端查询服务的结果多两个默认服务 (UUID: 0x1800 和 0x1801), 这是正常现象。正因如此, 对于同一服务, XY-MWC320A Bluetooth LE 客户端查询的 <srv\_index> 值等于 XY-MWC320A Bluetooth LE 服务端查询的 <srv\_index> 值 + 2。例如上述示例中的服务 0xA002, 当前在 XY-MWC320A Bluetooth LE 客户端查询到的 <srv\_index> 为 3, 如果在 XY-MWC320A Bluetooth LE 服务端通过 AT+BLEGATTSSRV? 命令查询, 则 <srv\_index> 为 1。

## 9. XY-MWC320A Bluetooth LE 客户端发现特征值。

AT

```
1 AT+BLEGATTCCHAR=0,3
```

Plain Text

```
1 +BLEGATTCCCHAR:"char",0,3,1,0xC300,0x02
2 +BLEGATTCCCHAR:"desc",0,3,1,1,0x2901
3 +BLEGATTCCCHAR:"char",0,3,2,0xC301,0x02
4 +BLEGATTCCCHAR:"desc",0,3,2,1,0x2901
5 +BLEGATTCCCHAR:"char",0,3,3,0xC302,0x08
6 +BLEGATTCCCHAR:"desc",0,3,3,1,0x2901
7 +BLEGATTCCCHAR:"char",0,3,4,0xC303,0x04
8 +BLEGATTCCCHAR:"desc",0,3,4,1,0x2901
9 +BLEGATTCCCHAR:"char",0,3,5,0xC304,0x08
10 +BLEGATTCCCHAR:"char",0,3,6,0xC305,0x10
11 +BLEGATTCCCHAR:"desc",0,3,6,1,0x2902
12 +BLEGATTCCCHAR:"char",0,3,7,0xC306,0x20
13 +BLEGATTCCCHAR:"desc",0,3,7,1,0x2902
14 +BLEGATTCCCHAR:"char",0,3,8,0xC307,0x02
15 +BLEGATTCCCHAR:"desc",0,3,8,1,0x2901
16 OK
```

## 10. XY-MWC320A Bluetooth LE 客户端读取一个特征值。

AT

```
1 AT+BLEGATTCRD=0,3,1
```

Plain Text

```
1 +BLEGATTCRD:0,1,0
2 OK
```

**说明：** 请注意目标特征值必须要有读权限。如果 XY-MWC320A Bluetooth LE 客户端读取特征成功，XY-MWC320A Bluetooth LE 服务端则会提示 +READ:0,"7c:df:a1:b3:8d:de"。

## 11. XY-MWC320A Bluetooth LE 客户端写一个特征值。

AT

```
1 AT+BLEGATTCWR=0,3,3,,2
```

Plain Text

```
1 >
```

符号 > 表示 AT 准备好接收串口数据，此时可以输入数据，当数据长度达到参数 <length> 的值时，执行写入操作。

Plain Text

```
1 OK
```

**说明：** 如果 XY-MWC320A Bluetooth LE 客户端写特征描述符成功，XY-MWC320A Bluetooth LE 服务端则会提示 +WRITE:<conn\_index>,<srv\_index>,<char\_index>,[<desc\_index>],<len>,<value>。

## 12. Indicate 一个特征值。

XY-MWC320A Bluetooth LE 客户端：

AT

1 AT+BLEGATTCWR=0,3,7,1,2

Plain Text

1 >

符号 > 表示 AT 准备好接收串口数据，此时可以输入数据，当数据长度达到参数 <length> 的值时，执行写入操作。

为了接收 XY-MWC320A Bluetooth LE 服务端发送过来的数据（通过 notify 方式或者 indicate 方式），XY-MWC320A Bluetooth LE 客户端需要提前向服务端注册。对于 notify 方式，需要写入值 0x0001，对于 indicate 方式，需要写入值 0x0002。在本例中写入 0x0002 来使用 indicate 方式。

Plain Text

1 OK

**说明：**如果 XY-MWC320A Bluetooth LE 客户端写特征描述符成功，XY-MWC320A Bluetooth LE 服务端则会提示 +WRITE:<conn\_index>,<srv\_index>,<char\_index>,<desc\_index>,<len>,<value>。XY-MWC320A Bluetooth LE 服务端：

AT

1 AT+BLEGATTSSIND=0,1,7,3

Plain Text

1 >

符号 > 表示 AT 准备好接收串口数据，此时可以输入数据，当数据长度达到参数 <length> 的值时，执行 indicate 操作。

Plain Text

1 OK

**说明：**如果 XY-MWC320A Bluetooth LE 客户端接收到 indication，则会提示 +INDICATE:<conn\_index>,<srv\_index>,<char\_index>,<len>,<value>。对于同一服务，XY-MWC320A Bluetooth LE 客户端的 <srv\_index> 值等于 XY-MWC320A Bluetooth LE 服务端的 <srv\_index> 值 + 2，这是正常现象。对于服务中特征的权限，可参考文档 如何自定义低功耗蓝牙服务。

### 2.4.3 Bluetooth LE 服务端读写服务特征值

以下示例同时使用两块 XY-MWC320A 开发板，其中一块作为 Bluetooth LE 服务端（只作为 Bluetooth LE 服务端角色），另一块作为 Bluetooth LE 客户端（只作为 Bluetooth LE 客户端角色）。这个例子展示了应如何建立 Bluetooth LE 连接，以及服务端读写服务特征值和客户端设置，notify 服务特征值。

## 重要

步骤中以 XY-MWC320A Bluetooth LE 服务端 开头的操作只需要在 XY-MWC320A Bluetooth LE 服务端执行即可，以 XY-MWC320A Bluetooth LE 客户端 开头的操作只需要在 XY-MWC320A Bluetooth LE 客户端执行即可。

### 1. 初始化 Bluetooth LE 功能。

XY-MWC320A Bluetooth LE 服务端：

AT

1 AT+BLEINIT=2

Plain Text

1 OK

XY-MWC320A Bluetooth LE 客户端：

AT

1 AT+BLEINIT=1

Plain Text

1 OK

### 2. XY-MWC320A Bluetooth LE 服务端创建服务。

AT

1 AT+BLEGATTSSRVCRE

Plain Text

1 OK

### 3. XY-MWC320A Bluetooth LE 服务端开启服务。

AT

1 AT+BLEGATTSSRVSTART

Plain Text

1 OK

### 4. XY-MWC320A Bluetooth LE 服务端获取其 MAC 地址。

AT

1 AT+BLEADDR?

Plain Text

1 +BLEADDR:"24:0a:c4:d6:e4:46"

2 OK

**说明：**查询到的地址可能与上述响应中的不同，请记住该地址，后续步骤中会用到。

## 5. XY-MWC320A Bluetooth LE 服务端设置广播参数。

AT

1 AT+BLEADVPARAM=50,50,0,0,7,0,,

Plain Text

1 OK

## 6. XY-MWC320A Bluetooth LE 服务端设置广播数据。

AT

1 AT+BLEADVDATA="0201060A09457370726573736966030302A0"

Plain Text

1 OK

## 7. XY-MWC320A Bluetooth LE 服务端开始广播。

AT

1 AT+BLEADVSTART

Plain Text

1 OK

## 8. XY-MWC320A Bluetooth LE 客户端创建服务。

AT

1 AT+BLEGATTSSRVCRE

Plain Text

1 OK

## 9. XY-MWC320A Bluetooth LE 客户端开启服务。

AT

1 AT+BLEGATTSSRVSTART

Plain Text

1 OK

## 10. XY-MWC320A Bluetooth LE 客户端获取其 MAC 地址。

AT

1 AT+BLEADDR?

Plain Text

```
1 +BLEADDR:"24:0a:c4:03:a7:4e"
2 OK
```

**说明：** 查询到的地址可能与上述响应中的不同，请记住该地址，后续步骤中会用到。

## 11. XY-MWC320A Bluetooth LE 客户端开始扫描，持续 3 秒。

AT

```
1 AT+BLESCAN=1,3
```

Plain Text

```
1 OK
2 +BLESCAN:"24:0a:c4:d6:e4:46",-78,0201060a09457370726573736966030302a0,,0
3 +BLESCAN:"45:03:cb:ac:aa:a0",-62,0201060aff4c001005441c61df7d,,1
4 +BLESCAN:"24:0a:c4:d6:e4:46",-26,0201060a09457370726573736966030302a0,,0
```

**说明：** 的扫描结果可能与上述响应中的不同。

## 12. 建立 the Bluetooth LE 连接。

XY-MWC320A Bluetooth LE 客户端：

AT

```
1 AT+BLECONN=0,"24:0a:c4:d6:e4:46"
```

Plain Text

```
1 +BLECONN:0,"24:0a:c4:d6:e4:46"
2 OK
```

**说明：** 输入上述命令时，请使用的 XY-MWC320A Bluetooth LE 服务端地址。如果 Bluetooth LE 连接成功，则会提示 +BLECONN:0,"24:0a:c4:d6:e4:46"。如果 Bluetooth LE 连接失败，则会提示 +BLECONN:0,-1。

XY-MWC320A Bluetooth LE 服务端：

AT

```
1 AT+BLECONN=0,"24:0a:c4:03:a7:4e"
```

Plain Text

```
1 +BLECONN:0,"24:0a:c4:03:a7:4e"
2 OK
```

- 输入上述命令时，请使用的 XY-MWC320A Bluetooth LE 客户端地址。
- 如果 Bluetooth LE 连接成功，则会提示 OK，不会提示 +BLECONN:0,"24:0a:c4:03:a7:4e"。
- 如果 Bluetooth LE 连接失败，则会提示 ERROR，不会提示 +BLECONN:0,-1。

### 1. XY-MWC320A Bluetooth LE 客户端查询本地服务。

AT

1 AT+BLEGATTSSRV?

Plain Text

```
1 +BLEGATTSSRV:1,1,0xA002,1
2 +BLEGATTSSRV:2,1,0xA003,1
3 OK
```

### 2. XY-MWC320A Bluetooth LE 客户端发现本地特征。

AT

1 AT+BLEGATTCHAR?

Plain Text

```
1 +BLEGATTCHAR:"char",1,1,0xC300,0x02
2 +BLEGATTCHAR:"desc",1,1,1,0x2901
3 +BLEGATTCHAR:"char",1,2,0xC301,0x02
4 +BLEGATTCHAR:"desc",1,2,1,0x2901
5 +BLEGATTCHAR:"char",1,3,0xC302,0x08
6 +BLEGATTCHAR:"desc",1,3,1,0x2901
7 +BLEGATTCHAR:"char",1,4,0xC303,0x04
8 +BLEGATTCHAR:"desc",1,4,1,0x2901
9 +BLEGATTCHAR:"char",1,5,0xC304,0x08
10 +BLEGATTCHAR:"char",1,6,0xC305,0x10
11 +BLEGATTCHAR:"desc",1,6,1,0x2902
12 +BLEGATTCHAR:"char",1,7,0xC306,0x20
13 +BLEGATTCHAR:"desc",1,7,1,0x2902
14 +BLEGATTCHAR:"char",1,8,0xC307,0x02
15 +BLEGATTCHAR:"desc",1,8,1,0x2901
16 +BLEGATTCHAR:"char",2,1,0xC400,0x02
17 +BLEGATTCHAR:"desc",2,1,1,0x2901
18 +BLEGATTCHAR:"char",2,2,0xC401,0x02
19 +BLEGATTCHAR:"desc",2,2,1,0x2901
20 OK
```

### 3. XY-MWC320A Bluetooth LE 服务端发现对端服务。

AT

1 AT+BLEGATTCPRIMSRV=0

Plain Text

```
1 +BLEGATTCPRIMSRV:0,1,0x1801,1
2 +BLEGATTCPRIMSRV:0,2,0x1800,1
3 +BLEGATTCPRIMSRV:0,3,0xA002,1
4 +BLEGATTCPRIMSRV:0,4,0xA003,1
5 OK
```

**说明：**XY-MWC320A Bluetooth LE 服务端查询服务的结果，比 XY-MWC320A Bluetooth LE 客户端查询服务的结果多两个默认服务（UUID: 0x1800 和 0x1801）。正因如此，对于同一服务，XY-MWC320A Bluetooth LE 服务端查询的 <srv\_index> 值等于 XY-MWC320A Bluetooth LE 客户端查询的 <srv\_index> 值 + 2。例如，上述示例中的服务 0xA002，当前在 XY-MWC320A Bluetooth LE 服务端查询到的 <srv\_index> 为 3，如果在 XY-MWC320A Bluetooth LE 服务端通过 AT+BLEGATTSSRV? 命令查询，则 <srv\_index> 为 1。

#### 4. XY-MWC320A Bluetooth LE 服务端发现对端特征。

AT

```
1 AT+BLEGATTCCHAR=0,3
```

Plain Text

```
1 +BLEGATTCCHAR:"char",0,3,1,0xC300,0x02
2 +BLEGATTCCHAR:"desc",0,3,1,1,0x2901
3 +BLEGATTCCHAR:"char",0,3,2,0xC301,0x02
4 +BLEGATTCCHAR:"desc",0,3,2,1,0x2901
5 +BLEGATTCCHAR:"char",0,3,3,0xC302,0x08
6 +BLEGATTCCHAR:"desc",0,3,3,1,0x2901
7 +BLEGATTCCHAR:"char",0,3,4,0xC303,0x04
8 +BLEGATTCCHAR:"desc",0,3,4,1,0x2901
9 +BLEGATTCCHAR:"char",0,3,5,0xC304,0x08
10 +BLEGATTCCHAR:"char",0,3,6,0xC305,0x10
11 +BLEGATTCCHAR:"desc",0,3,6,1,0x2902
12 +BLEGATTCCHAR:"char",0,3,7,0xC306,0x20
13 +BLEGATTCCHAR:"desc",0,3,7,1,0x2902
14 +BLEGATTCCHAR:"char",0,3,8,0xC307,0x02
15 +BLEGATTCCHAR:"desc",0,3,8,1,0x2901
16 OK
```

#### 5. XY-MWC320A Bluetooth LE 客户端设置服务特征值。

选择支持写操作的服务特征 (characteristic) 去设置服务特征值。

AT

```
1 AT+BLEGATTSSETATTR=1,8,,1
```

Plain Text

```
1 >
```

## 写入一个字节 ``9``

Plain Text

1 OK

## 6. XY-MWC320A Bluetooth LE 服务端读服务特征值。

AT

1 AT+BLEGATTCDR=0,3,8,

Plain Text

1 +BLEGATTCDR:0,1,9

2 OK

## 7. XY-MWC320A Bluetooth LE 服务端写服务特征值。

选择支持写操作的服务特性写入特性。

AT

1 AT+BLEGATTCWR=0,3,6,1,2

Plain Text

1 >

## 写入2个字节 ``12``

Plain Text

1 OK

**说明：**如果 Bluetooth LE 服务端写服务特征值成功后，Bluetooth LE 客户端则会提示 +WRITE:0,1,6,1,2,12。

## 8. XY-MWC320A Bluetooth LE 客户端 notify 服务特征值

AT

1 AT+BLEGATTSNTFY=0,1,6,10

Plain Text

1 >

## 写入 ``1234567890`` 10个字节

Plain Text

1 OK

**说明：**如果 XY-MWC320A Bluetooth LE 客户端 notify 服务特征值给服务端成功，Bluetooth LE 服务端则会提示 +NOTIFY:0,3,6,10,1234567890。

## 2.4.4 Bluetooth LE 连接加密

以下示例同时使用两块 XY-MWC320A 开发板，其中一块作为 Bluetooth LE 服务端（只作为 Bluetooth LE 服务端角色），另一块作为 Bluetooth LE 客户端（只作为 Bluetooth LE 客户端角色）。这个例子展示了怎样加密 Bluetooth LE 连接。

### 重要

- 在以下步骤中以 XY-MWC320A Bluetooth LE 服务端 开头的操作只需要在 XY-MWC320A Bluetooth LE 服务端执行即可，以 XY-MWC320A Bluetooth LE 客户端 开头的操作只需要在 XY-MWC320A Bluetooth LE 客户端执行即可。
- 加密 和 绑定 是两个不同的概念。绑定 只是加密成功后在本地存储了一个长期的密钥。
- XY-MWC320A 最多允许绑定 10 个设备。

### 1. 初始化 Bluetooth LE 功能。

XY-MWC320A Bluetooth LE 服务端：

```
AT
1 AT+BLEINIT=2
```

```
Plain Text
1 OK
```

XY-MWC320A Bluetooth LE 客户端：

```
AT
1 AT+BLEINIT=1
```

```
Plain Text
1 OK
```

### 2. XY-MWC320A Bluetooth LE 服务端获取 Bluetooth LE 地址。

```
AT
1 AT+BLEADDR?
```

```
Plain Text
1 +BLEADDR:"24:0a:c4:d6:e4:46"
2 OK
```

**说明：**查询到的地址可能与上述响应中的不同，请记住该地址，后续步骤中会用到。

### 3. XY-MWC320A Bluetooth LE 服务端创建服务。

```
AT
1 AT+BLEGATTSSRVCRE
```

Plain Text

1 OK

#### 4. XY-MWC320A Bluetooth LE 服务端开启服务。

AT

1 AT+BLEGATTSSRVSTART

Plain Text

1 OK

#### 5. XY-MWC320A Bluetooth LE 服务端发现服务特征。

AT

1 AT+BLEGATTSCCHAR?

Plain Text

```
1 +BLEGATTSCCHAR:"char",1,1,0xC300,0x02
2 +BLEGATTSCCHAR:"desc",1,1,1,0x2901
3 +BLEGATTSCCHAR:"char",1,2,0xC301,0x02
4 +BLEGATTSCCHAR:"desc",1,2,1,0x2901
5 +BLEGATTSCCHAR:"char",1,3,0xC302,0x08
6 +BLEGATTSCCHAR:"desc",1,3,1,0x2901
7 +BLEGATTSCCHAR:"char",1,4,0xC303,0x04
8 +BLEGATTSCCHAR:"desc",1,4,1,0x2901
9 +BLEGATTSCCHAR:"char",1,5,0xC304,0x08
10 +BLEGATTSCCHAR:"char",1,6,0xC305,0x10
11 +BLEGATTSCCHAR:"desc",1,6,1,0x2902
12 +BLEGATTSCCHAR:"char",1,7,0xC306,0x20
13 +BLEGATTSCCHAR:"desc",1,7,1,0x2902
14 +BLEGATTSCCHAR:"char",1,8,0xC307,0x02
15 +BLEGATTSCCHAR:"desc",1,8,1,0x2901
16 +BLEGATTSCCHAR:"char",2,1,0xC400,0x02
17 +BLEGATTSCCHAR:"desc",2,1,1,0x2901
18 +BLEGATTSCCHAR:"char",2,2,0xC401,0x02
19 +BLEGATTSCCHAR:"desc",2,2,1,0x2901
20 OK
```

#### 6. XY-MWC320A Bluetooth LE 服务端开始广播，之后 XY-MWC320A Bluetooth LE 客户端开始扫描并且持续 3 秒钟。

XY-MWC320A Bluetooth LE 服务端：

AT

1 AT+BLEADVSTART

Plain Text

1 OK

## XY-MWC320A Bluetooth LE 客户端:

AT

1 AT+BLESCAN=1,3

Plain Text

1 OK  
2 +BLESCAN:"5b:3b:6c:51:90:49",-87,02011a020a0c0aff4c001005071c3024dc,,1  
3 +BLESCAN:"c4:5b:be:93:ec:66",-84,0201060303111809095647543147572d58020a03,,0  
4 +BLESCAN:"24:0a:c4:d6:e4:46",-29,,,0

**说明:** 的扫描结果可能与上述响应中的不同。

## 7. 建立 Bluetooth LE 连接。

### XY-MWC320A Bluetooth LE 客户端:

AT

1 AT+BLECONN=0,"24:0a:c4:d6:e4:46"

Plain Text

1 +BLECONN:0,"24:0a:c4:d6:e4:46"  
2 OK

**说明:** 输入上述命令时, 请使用的 XY-MWC320A Bluetooth LE 服务端地址。如果 Bluetooth LE 连接成功, 则会提示 +BLECONN:0,"24:0a:c4:d6:e4:46"。如果 Bluetooth LE 连接失败, 则会提示 +BLECONN:0,-1。

## 8. XY-MWC320A Bluetooth LE 客户端发现服务。

AT

1 AT+BLEGATTCPRIMSRV=0

Plain Text

1 +BLEGATTCPRIMSRV:0,1,0x1801,1  
2 +BLEGATTCPRIMSRV:0,2,0x1800,1  
3 +BLEGATTCPRIMSRV:0,3,0xA002,1  
4 +BLEGATTCPRIMSRV:0,4,0xA003,1  
5 OK

**说明:** XY-MWC320A Bluetooth LE 客户端查询服务的结果, 比 XY-MWC320A Bluetooth LE 服务端查询服务的结果多两个默认服务 (UUID: 0x1800 和 0x1801), 这是正常现象。正因如此, 对于同一服务, XY-MWC320A Bluetooth LE 客户端查询的 <srv\_index> 值等于 XY-MWC320A Bluetooth LE 服务端查询的 <srv\_index> 值 + 2。例如上述示例中的服务 0xA002, 当前在 XY-MWC320A Bluetooth LE 客户

端查询到的 <srv\_index> 为 3, 如果在 XY-MWC320A Bluetooth LE 服务端通过 AT+BLEGATTSSRV? 命令查询, 则 <srv\_index> 为 1。

## 9. XY-MWC320A Bluetooth LE 客户端发现特征值。

AT

```
1 AT+BLEGATTCCHAR=0,3
```

Plain Text

```
1 +BLEGATTCCHAR:"char",0,3,1,0xC300,0x02
2 +BLEGATTCCHAR:"desc",0,3,1,1,0x2901
3 +BLEGATTCCHAR:"char",0,3,2,0xC301,0x02
4 +BLEGATTCCHAR:"desc",0,3,2,1,0x2901
5 +BLEGATTCCHAR:"char",0,3,3,0xC302,0x08
6 +BLEGATTCCHAR:"desc",0,3,3,1,0x2901
7 +BLEGATTCCHAR:"char",0,3,4,0xC303,0x04
8 +BLEGATTCCHAR:"desc",0,3,4,1,0x2901
9 +BLEGATTCCHAR:"char",0,3,5,0xC304,0x08
10 +BLEGATTCCHAR:"char",0,3,6,0xC305,0x10
11 +BLEGATTCCHAR:"desc",0,3,6,1,0x2902
12 +BLEGATTCCHAR:"char",0,3,7,0xC306,0x20
13 +BLEGATTCCHAR:"desc",0,3,7,1,0x2902
14 +BLEGATTCCHAR:"char",0,3,8,0xC307,0x02
15 +BLEGATTCCHAR:"desc",0,3,8,1,0x2901
16 OK
```

## 10. 设置加密参数。设置 auth\_req 为 SC\_MITM\_BOND, 服务端的 iocap 为 KeyboardOnly, 客户端的 iocap 为 KeyboardDisplay, key\_size 为 16, init\_key 为 3, rsp\_key 为 3。

XY-MWC320A Bluetooth LE 服务端:

AT

```
1 AT+BLESECPARAM=13,2,16,3,3
```

Plain Text

```
1 OK
```

XY-MWC320A Bluetooth LE 客户端:

AT

```
1 AT+BLESECPARAM=13,4,16,3,3
```

Plain Text

```
1 OK
```

**说明:** 在本例中, XY-MWC320A Bluetooth LE 服务端输入配对码, XY-MWC320A Bluetooth LE 客户端显示配对码。XY-MWC320A 支持 Legacy Pairing 和 Secure Connections 两种加密方式, 但后者有更

高级别的优先级。如果对端也支持 Secure Connections，则会采用 Secure Connections 方式加密。

## 11. XY-MWC320A Bluetooth LE 客户端发起加密请求。

AT

1 AT+BLEENC=0,3

Plain Text

1 OK

**说明：**如果 XY-MWC320A Bluetooth LE 服务端成功接收到加密请求，XY-MWC320A Bluetooth LE 服务端则会提示 +BLESECREQ:0。

## 12. XY-MWC320A Bluetooth LE 服务端响应配对请求。

AT

1 AT+BLEENCRSP=0,1

Plain Text

1 OK

- 如果 XY-MWC320A Bluetooth LE 客户端成功收到配对响应，则 XY-MWC320A Bluetooth LE 客户端将会产生一个 6 位的配对码。
- 在本例中，XY-MWC320A Bluetooth LE 客户端则会提示 +BLESECNTFYKEY:0,793718。配对码为 793718。

### 1. XY-MWC320A Bluetooth LE 客户端回复配对码。

AT

1 AT+BLEKEYREPLY=0,793718

Plain Text

1 OK

执行这个命令之后，在 XY-MWC320A Bluetooth LE 服务端和 XY-MWC320A Bluetooth LE 客户端会有一些对应信息提示。

XY-MWC320A Bluetooth LE 服务端：

Plain Text

```
1 +BLESECKEYTYPE:0,16
2 +BLESECKEYTYPE:0,1
3 +BLESECKEYTYPE:0,32
4 +BLESECKEYTYPE:0,2
5 +BLEAUTHCMPL:0,0
```

XY-MWC320A Bluetooth LE 客户端：

Plain Text

```
1 +BLESECNTFYKEY:0,793718
2 +BLESECKEYTYPE:0,2
3 +BLESECKEYTYPE:0,16
4 +BLESECKEYTYPE:0,1
5 +BLESECKEYTYPE:0,32
6 +BLEAUTHCMPL:0,0
```

可以忽略以 +BLESECKEYTYPE 开头的信息。信息 +BLEAUTHCMPL:0,0 中的第二个参数为 0 表示加密成功，为 1 表示加密失败。

### 2.4.5 两个 XY-MWC320A 开发板之间建立 SPP 连接，以及在 UART-Bluetooth LE 透传模式下传输数据

以下示例同时使用两块 XY-MWC320A 开发板，其中一块作为 Bluetooth LE 服务端（只作为 Bluetooth LE 服务端角色），另一块作为 Bluetooth LE 客户端（只作为 Bluetooth LE 客户端角色）。这个例子展示了应如何建立 Bluetooth LE 连接，以及建立透传通信 Bluetooth LE SPP (Serial Port Profile, UART-Bluetooth LE 透传模式)。

#### 重要

在以下步骤中以 XY-MWC320A Bluetooth LE 服务端 开头的操作只需要在 XY-MWC320A Bluetooth LE 服务端执行即可，以 XY-MWC320A Bluetooth LE 客户端 开头的操作只需要在 XY-MWC320A Bluetooth LE 客户端执行即可。

#### 1. 初始化 Bluetooth LE 功能。

XY-MWC320A Bluetooth LE 服务端：

AT

```
1 AT+BLEINIT=2
```

Plain Text

```
1 OK
```

XY-MWC320A Bluetooth LE 客户端：

AT

```
1 AT+BLEINIT=1
```

Plain Text

```
1 OK
```

#### 2. XY-MWC320A Bluetooth LE 服务端创建服务。

AT

```
1 AT+BLEGATTSSRVCRE
```

Plain Text

1 OK

### 3. XY-MWC320A Bluetooth LE 服务端开启服务。

AT

1 AT+BLEGATTSSRVSTART

Plain Text

1 OK

### 4. XY-MWC320A Bluetooth LE 服务端获取其 MAC 地址。

AT

1 AT+BLEADDR?

Plain Text

1 +BLEADDR:"24:0a:c4:d6:e4:46"

2 OK

**说明：**查询到的地址可能与上述响应中的不同，请记住该地址，后续步骤中会用到。

### 5. XY-MWC320A Bluetooth LE 服务端设置广播参数。

AT

1 AT+BLEADVPARAM=50,50,0,0,7,0,,

Plain Text

1 OK

### 6. XY-MWC320A Bluetooth LE 服务端设置广播数据。

AT

1 AT+BLEADVDATA="0201060A09457370726573736966030302A0"

Plain Text

1 OK

### 7. XY-MWC320A Bluetooth LE 服务端开始广播。

AT

1 AT+BLEADVSTART

Plain Text

1 OK

### 8. XY-MWC320A Bluetooth LE 客户端开始扫描，持续 3 秒。

AT

1 AT+BLESCAN=1,3

Plain Text

1 OK

2 +BLESCAN:"24:0a:c4:d6:e4:46",-78,0201060a09457370726573736966030302a0,,0

3 +BLESCAN:"45:03:cb:ac:aa:a0",-62,0201060aff4c001005441c61df7d,,1

4 +BLESCAN:"24:0a:c4:d6:e4:46",-26,0201060a09457370726573736966030302a0,,0

**说明：**的扫描结果可能与上述响应中的不同。

## 9. 建立 the Bluetooth LE 连接。

XY-MWC320A Bluetooth LE 客户端：

AT

1 AT+BLECONN=0,"24:0a:c4:d6:e4:46"

Plain Text

1 +BLECONN:0,"24:0a:c4:d6:e4:46"

2 OK

**说明：**输入上述命令时，请使用的 XY-MWC320A Bluetooth LE 服务端地址。如果 Bluetooth LE 连接成功，则会提示 +BLECONN:0,"24:0a:c4:d6:e4:46"。如果 Bluetooth LE 连接失败，则会提示 +BLECONN:0,-1。

## 10. XY-MWC320A Bluetooth LE 服务端查询服务。

AT

1 AT+BLEGATTSSRV?

Plain Text

1 +BLEGATTSSRV:1,1,0xA002,1

2 +BLEGATTSSRV:2,1,0xA003,1

3 OK

## 11. XY-MWC320A Bluetooth LE 服务端发现特征。

AT

1 AT+BLEGATTCHAR?

Plain Text

```
1 +BLEGATTSCCHAR:"char",1,1,0xC300,0x02
2 +BLEGATTSCCHAR:"desc",1,1,1,0x2901
3 +BLEGATTSCCHAR:"char",1,2,0xC301,0x02
4 +BLEGATTSCCHAR:"desc",1,2,1,0x2901
5 +BLEGATTSCCHAR:"char",1,3,0xC302,0x08
6 +BLEGATTSCCHAR:"desc",1,3,1,0x2901
7 +BLEGATTSCCHAR:"char",1,4,0xC303,0x04
8 +BLEGATTSCCHAR:"desc",1,4,1,0x2901
9 +BLEGATTSCCHAR:"char",1,5,0xC304,0x08
10 +BLEGATTSCCHAR:"char",1,6,0xC305,0x10
11 +BLEGATTSCCHAR:"desc",1,6,1,0x2902
12 +BLEGATTSCCHAR:"char",1,7,0xC306,0x20
13 +BLEGATTSCCHAR:"desc",1,7,1,0x2902
14 +BLEGATTSCCHAR:"char",1,8,0xC307,0x02
15 +BLEGATTSCCHAR:"desc",1,8,1,0x2901
16 +BLEGATTSCCHAR:"char",2,1,0xC400,0x02
17 +BLEGATTSCCHAR:"desc",2,1,1,0x2901
18 +BLEGATTSCCHAR:"char",2,2,0xC401,0x02
19 +BLEGATTSCCHAR:"desc",2,2,1,0x2901
20 OK
```

## 12. XY-MWC320A Bluetooth LE 客户端发现服务。

AT

```
1 AT+BLEGATTCPRIMSRV=0
```

Plain Text

```
1 +BLEGATTCPRIMSRV:0,1,0x1801,1
2 +BLEGATTCPRIMSRV:0,2,0x1800,1
3 +BLEGATTCPRIMSRV:0,3,0xA002,1
4 +BLEGATTCPRIMSRV:0,4,0xA003,1
5 OK
```

**说明：**XY-MWC320A Bluetooth LE 客户端查询服务的结果，比 XY-MWC320A Bluetooth LE 服务端查询服务的结果多两个默认服务（UUID: 0x1800 和 0x1801），这是正常现象。正因如此，对于同一服务，XY-MWC320A Bluetooth LE 客户端查询的 <srv\_index> 值等于 XY-MWC320A Bluetooth LE 服务端查询的 <srv\_index> 值 + 2。例如，上述示例中的服务 0xA002，当前在 XY-MWC320A Bluetooth LE 客户端查询到的 <srv\_index> 为 3，如果在 XY-MWC320A Bluetooth LE 服务端通过 AT+BLEGATTSSRV? 命令查询，则 <srv\_index> 为 1。

## 13. XY-MWC320A Bluetooth LE 客户端发现特征。

AT

```
1 AT+BLEGATTCCHAR=0,3
```

Plain Text

```
1 +BLEGATTCCCHAR:"char",0,3,1,0xC300,0x02
2 +BLEGATTCCCHAR:"desc",0,3,1,1,0x2901
3 +BLEGATTCCCHAR:"char",0,3,2,0xC301,0x02
4 +BLEGATTCCCHAR:"desc",0,3,2,1,0x2901
5 +BLEGATTCCCHAR:"char",0,3,3,0xC302,0x08
6 +BLEGATTCCCHAR:"desc",0,3,3,1,0x2901
7 +BLEGATTCCCHAR:"char",0,3,4,0xC303,0x04
8 +BLEGATTCCCHAR:"desc",0,3,4,1,0x2901
9 +BLEGATTCCCHAR:"char",0,3,5,0xC304,0x08
10 +BLEGATTCCCHAR:"char",0,3,6,0xC305,0x10
11 +BLEGATTCCCHAR:"desc",0,3,6,1,0x2902
12 +BLEGATTCCCHAR:"char",0,3,7,0xC306,0x20
13 +BLEGATTCCCHAR:"desc",0,3,7,1,0x2902
14 +BLEGATTCCCHAR:"char",0,3,8,0xC307,0x02
15 +BLEGATTCCCHAR:"desc",0,3,8,1,0x2901
16 OK
```

#### 14. XY-MWC320A Bluetooth LE 客户端配置 Bluetooth LE SPP。

选择支持写操作的服务特征 (characteristic) 作为写通道发送数据，选择支持 notify 或者 indicate 的 characteristic 作为读通道接收数据。

AT

```
1 AT+BLESPPCFG=1,3,5,3,7
```

Plain Text

```
1 OK
```

#### 15. XY-MWC320A Bluetooth LE 客户端使能 Bluetooth LE SPP。

AT

```
1 AT+BLESPP
```

Plain Text

```
1 OK
2 >
```

上述响应表示 AT 已经进入 Bluetooth LE SPP 模式，可以进行数据的发送和接收。

**说明：**XY-MWC320A Bluetooth LE 客户端开启 Bluetooth LE SPP 透传模式后，串口收到的数据会通过 Bluetooth LE 传输到 XY-MWC320A Bluetooth LE 服务端。

#### 16. XY-MWC320A Bluetooth LE 服务端配置 Bluetooth LE SPP。

选择支持 notify 或者 indicate 的 characteristic 作为写通道发送数据，选择支持写操作的 characteristic 作为读通道接收数据。

AT

1 AT+BLESPPCFG=1,1,7,1,5

Plain Text

1 OK

## 17. XY-MWC320A Bluetooth LE 服务端使能 Bluetooth LE SPP。

AT

1 AT+BLESPP

Plain Text

1 OK

2 >

上述响应表示 AT 已经进入 Bluetooth LE SPP 模式，可以进行数据的发送和接收。

**说明：**XY-MWC320A Bluetooth LE 服务端开启 Bluetooth LE SPP 透传模式后，串口收到的数据会通过 Bluetooth LE 传输到 XY-MWC320A Bluetooth LE 客户端。如果 XY-MWC320A Bluetooth LE 客户端没有先开启 Bluetooth LE SPP 透传，或者使用其他设备作为 Bluetooth LE 客户端，则 XY-MWC320A Bluetooth LE 客户端需要先开启侦听 Notify 或者 Indicate。例如，XY-MWC320A Bluetooth LE 客户端如果未开启透传，则应先调用 AT+BLEGATTCWR=0,3,7,1,1 开启侦听，XY-MWC320A Bluetooth LE 服务端 才能成功实现透传。对于同一服务，XY-MWC320A Bluetooth LE 客户端的 <srv\_index> 值等于 XY-MWC320A Bluetooth LE 服务端的 <srv\_index> 值 + 2，这是正常现象。

### 2.4.6 XY-MWC320A 与手机建立 SPP 连接，以及在 UART-Bluetooth LE 透传模式下传输数据

该示例展示了如何在 XY-MWC320A 开发板（仅作为低功耗蓝牙服务器角色）和手机（仅作为低功耗蓝牙客户端角色）之间建立 SPP 连接，以及如何在 UART-Bluetooth LE 透传模式下传输数据。

#### 重要

步骤中以 XY-MWC320A Bluetooth LE 服务端 开头的操作只需要在 XY-MWC320A Bluetooth LE 服务端执行即可，而以 Bluetooth LE 客户端 开头的操作只需要在手机的蓝牙调试助手中执行即可。

1. 在手机端下载 Bluetooth LE 调试助手，例如 LightBlue。

2. 初始化 Bluetooth LE 功能。

XY-MWC320A Bluetooth LE 服务端：

AT

1 AT+BLEINIT=2

Plain Text

1 OK

### 3. XY-MWC320A Bluetooth LE 服务端创建服务。

AT

1 AT+BLEGATTSSRVCRE

Plain Text

1 OK

#### 4. XY-MWC320A Bluetooth LE 服务端开启服务。

AT

1 AT+BLEGATTSSRVSTART

Plain Text

1 OK

#### 5. XY-MWC320A Bluetooth LE 服务端获取其 MAC 地址。

AT

1 AT+BLEADDR?

Plain Text

1 +BLEADDR:"24:0a:c4:d6:e4:46"

2 OK

**说明：** 查询到的地址可能与上述响应中的不同，请记住该地址，后续步骤中会用到。

#### 6. XY-MWC320A Bluetooth LE 服务端设置广播参数。

AT

1 AT+BLEADVPARAM=50,50,0,0,7,0,,

Plain Text

1 OK

#### 7. XY-MWC320A Bluetooth LE 服务端设置广播数据。

AT

1 AT+BLEADVDATA="0201060A09457370726573736966030302A0"

Plain Text

1 OK

#### 8. XY-MWC320A Bluetooth LE 服务端开始广播。

AT

1 AT+BLEADVSTART

Plain Text

1 OK

## 9. 创建 Bluetooth LE 连接。

手机打开 LightBlue 应用程序，并打开 SCAN 开始扫描，找到 XY-MWC320A Bluetooth LE 服务端的 MAC 地址，点击 CONNECT 进行连接。此时 XY-MWC320A 端应该会打印类似于 +BLECONN:0,"60:51:42:fe:98:aa" 的日志，这表示已经建立了 Bluetooth LE 连接。

## 10. XY-MWC320A Bluetooth LE 服务端查询服务。

AT

1 AT+BLEGATTSSRV?

Plain Text

1 +BLEGATTSSRV:1,1,0xA002,1  
2 +BLEGATTSSRV:2,1,0xA003,1  
3 OK

## 11. XY-MWC320A Bluetooth LE 服务端发现特征。

AT

1 AT+BLEGATTCHAR?

Plain Text

1 +BLEGATTCHAR:"char",1,1,0xC300,0x02  
2 +BLEGATTCHAR:"desc",1,1,1,0x2901  
3 +BLEGATTCHAR:"char",1,2,0xC301,0x02  
4 +BLEGATTCHAR:"desc",1,2,1,0x2901  
5 +BLEGATTCHAR:"char",1,3,0xC302,0x08  
6 +BLEGATTCHAR:"desc",1,3,1,0x2901  
7 +BLEGATTCHAR:"char",1,4,0xC303,0x04  
8 +BLEGATTCHAR:"desc",1,4,1,0x2901  
9 +BLEGATTCHAR:"char",1,5,0xC304,0x08  
10 +BLEGATTCHAR:"char",1,6,0xC305,0x10  
11 +BLEGATTCHAR:"desc",1,6,1,0x2902  
12 +BLEGATTCHAR:"char",1,7,0xC306,0x20  
13 +BLEGATTCHAR:"desc",1,7,1,0x2902  
14 +BLEGATTCHAR:"char",1,8,0xC307,0x02  
15 +BLEGATTCHAR:"desc",1,8,1,0x2901  
16 +BLEGATTCHAR:"char",2,1,0xC400,0x02  
17 +BLEGATTCHAR:"desc",2,1,1,0x2901  
18 +BLEGATTCHAR:"char",2,2,0xC401,0x02  
19 +BLEGATTCHAR:"desc",2,2,1,0x2901  
20 OK

## 12. Bluetooth LE 客户端发现特征。

此时在手机 LightBlue 客户端选择点击 Properties 为 NOTIFY 或者 INDICATE 的服务特征（这里 XY-MWC320A 默认 Properties 为 NOTIFY 或者 INDICATE 的服务特征是 0xC305 和 0xC306），开始侦听 Properties 为 NOTIFY 或者 INDICATE 的服务特征。

## 13. XY-MWC320A Bluetooth LE 服务端配置 Bluetooth LE SPP。

选择支持 notify 或者 indicate 的 characteristic 作为写通道发送数据，选择支持写操作的 characteristic 作为读通道接收数据。

```
AT
1 AT+BLESPPCFG=1,1,7,1,5
```

```
Plain Text
1 OK
```

## 14. XY-MWC320A Bluetooth LE 服务端使能 Bluetooth LE SPP。

```
AT
1 AT+BLESPP
```

```
Plain Text
1 OK
2 >
```

上述响应表示 AT 已经进入 Bluetooth LE SPP 模式，可以进行数据的发送和接收。

## 15. Bluetooth LE 客户端发送数据。

在 LightBlue 客户端选择 0xC304 服务特征值发送数据 test 给 XY-MWC320A Bluetooth LE 服务端，此时 XY-MWC320A Bluetooth LE 服务端可以收到 test。

## 16. XY-MWC320A Bluetooth LE 服务端发送数据。

在 XY-MWC320A Bluetooth LE 服务端直接发送 test，此时 LightBlue 客户端可以收到 test。

### 2.4.7 XY-MWC320A 和手机之间建立 Bluetooth LE 连接并配对

该示例展示了如何在 XY-MWC320A 开发板（仅作为低功耗蓝牙服务器角色）和手机（仅作为低功耗蓝牙客户端角色）之间建立 Bluetooth LE 连接并输入密钥完成配对。

#### 重要

步骤中以 XY-MWC320A Bluetooth LE 服务端 开头的操作只需要在 XY-MWC320A Bluetooth LE 服务端执行即可，而以 Bluetooth LE 客户端 开头的操作只需要在手机的蓝牙调试助手中执行即可。

1. 在手机端下载 Bluetooth LE 调试助手，例如 LightBlue 应用程序。

2. 初始化 Bluetooth LE 功能。

## XY-MWC320A Bluetooth LE 服务端：

AT

1 AT+BLEINIT=2

Plain Text

1 OK

### 3. XY-MWC320A Bluetooth LE 服务端创建服务。

AT

1 AT+BLEGATTSSRVCRE

Plain Text

1 OK

### 4. XY-MWC320A Bluetooth LE 服务端开启服务。

AT

1 AT+BLEGATTSSRVSTART

Plain Text

1 OK

### 5. XY-MWC320A Bluetooth LE 服务端获取其 MAC 地址。

AT

1 AT+BLEADDR?

Plain Text

1 +BLEADDR:"24:0a:c4:d6:e4:46"

2 OK

**说明：** 查询到的地址可能与上述响应中的不同，请记住该地址，后续步骤中会用到。

### 6. XY-MWC320A Bluetooth LE 服务端设置广播参数。

AT

1 AT+BLEADVPARAM=50,50,0,0,7,0,,

Plain Text

1 OK

### 7. XY-MWC320A Bluetooth LE 服务端设置广播数据。

AT

1 AT+BLEADVDATA="0201060A09457370726573736966030302A0"

```
Plain Text
```

```
1 OK
```

## 8. XY-MWC320A Bluetooth LE 服务端设置加密参数。

```
AT
```

```
1 AT+BLESECPARAM=13,2,16,3,3
```

```
Plain Text
```

```
1 OK
```

## 9. XY-MWC320A Bluetooth LE 服务端开始广播。

```
AT
```

```
1 AT+BLEADVSTART
```

```
Plain Text
```

```
1 OK
```

## 10. Bluetooth LE 客户端创建连接。

手机打开 LightBlue 应用程序，并打开 SCAN 开始扫描，找到 XY-MWC320A Bluetooth LE 服务端的 MAC 地址，点击 CONNECT 进行连接。此时 XY-MWC320A 端应该会打印类似于 +BLECONN:0,"60:51:42:fe:98:aa" 的日志，这表示已经建立了 Bluetooth LE 连接。

## 11. XY-MWC320A Bluetooth LE 服务端发起加密请求。

```
AT
```

```
1 AT+BLEENC=0,3
```

```
Plain Text
```

```
1 OK
```

## 12. Bluetooth LE 客户端同意配对。

手机 LightBlue 应用程序刚刚创建成功的 Bluetooth LE 连接的页面会弹出配对信息（包含配对密钥，例如密钥为：231518），点击 配对。此时 XY-MWC320A 端应该会打印类似于 +BLESECKEYREQ:0 的日志，这表示手机已经响应配对。

## 13. XY-MWC320A Bluetooth LE 服务端回复配对密钥。

此时 Bluetooth LE 服务端回复的密钥即为上一步骤中手机 LightBlue 应用程序 弹出的配对信息中的密钥：231518。

```
AT
```

```
1 AT+BLEKEYREPLY=0,231518
```

Plain Text

1 OK

此时 XY-MWC320A Bluetooth LE 服务端会打印类似于以下日志，这表示 XY-MWC320A Bluetooth LE 服务端与手机 Bluetooth LE 客户端完成了配对。

Plain Text

```
1 +BLESECKEYTYPE:0,16
2 +BLESECKEYTYPE:0,1
3 +BLESECKEYTYPE:0,32
4 +BLESECKEYTYPE:0,2
5 +BLEAUTHCMPL:0,0
```

## 2.5 MQTT AT 示例

本文档主要提供在 XY-MWC320A 设备上运行 MQTT AT 命令集 命令的详细示例。

### 重要

文档中所描述的例子均基于设备已连接 Wi-Fi 的前提。

### 2.5.1 基于 TCP 的 MQTT 连接（需要本地创建 MQTT 代理）（适用于数据量少）

以下示例同时使用两块 XY-MWC320A 开发板，其中一块作为 MQTT 发布者（只作为 MQTT 发布者角色），另一块作为 MQTT 订阅者（只作为 MQTT 订阅者角色）。

示例介绍了如何基于 TCP 创建 MQTT 连接。首先需要创建一个本地 MQTT 代理，假设 MQTT 代理的 IP 地址为 192.168.3.102，端口为 8883。

### 重要

步骤中以 XY-MWC320A MQTT 发布者 开头的操作只需要在 XY-MWC320A MQTT 发布者端执行即可，以 XY-MWC320A MQTT 订阅者 开头的操作只需要在 XY-MWC320A MQTT 订阅者端执行即可。如果操作没有特别指明在哪端操作，则需要在发布者端和订阅者端都执行。

#### 1. 设置 MQTT 用户属性。

XY-MWC320A MQTT 发布者：

AT

```
1 AT+MQTTUSERCFG=0,1,"publisher","xyuser","123456789",0,0,""
```

Plain Text

1 OK

XY-MWC320A MQTT 订阅者：

AT

```
1 AT+MQTTUSERCFG=0,1,"subscriber","xyuser","123456789",0,0,""
```

Plain Text

1 OK

## 2. 连接 MQTT 代理。

AT

1 AT+MQTTCONN=0,"192.168.3.102",8883,1

Plain Text

1 +MQTTCONNECTED:0,1,"192.168.3.102","8883","",1

2 OK

**说明：**输入的 MQTT 代理域名或 MQTT 代理 IP 地址可能与上述命令中的不同。

## 3. 订阅 MQTT 主题。

XY-MWC320A MQTT 订阅者：

AT

1 AT+MQTTSUB=0,"topic",1

Plain Text

1 OK

## 4. 发布 MQTT 消息（字符串）。

XY-MWC320A MQTT 发布者：

AT

1 AT+MQTTPUB=0,"topic","test",1,0

Plain Text

1 OK

**说明：**如果 XY-MWC320A MQTT 发布者成功发布消息，以下信息将会在 XY-MWC320A MQTT 订阅者端提示。

Plain Text

1 +MQTTSUBRECV:0,"topic",4,test

- 如果 XY-MWC320A MQTT 发布者成功发布消息，以下信息将会在 XY-MWC320A MQTT 订阅者端提示。

Plain Text

1 +MQTTSUBRECV:0,"topic",4,test

## 1. 关闭 MQTT 连接。

AT

```
1 AT+MQTTCLEAN=0
```

Plain Text

```
1 OK
```

### 2.5.2 基于 TCP 的 MQTT 连接（需要本地创建 MQTT 代理）（适用于数据量多）

以下示例同时使用两块 XY-MWC320A 开发板，其中一块作为 MQTT 发布者（只作为 MQTT 发布者角色），另一块作为 MQTT 订阅者（只作为 MQTT 订阅者角色）。

示例介绍了如何基于 TCP 创建 MQTT 连接。首先需要创建一个本地 MQTT 代理，假设 MQTT 代理的 IP 地址为 192.168.3.102，端口为 8883。

如果发布消息的数据量相对较多，已经超过了单条 AT 命令的长度阈值 256，则可以使用 AT+MQTTPUBRAW 命令。

#### 重要

步骤中以 XY-MWC320A MQTT 发布者开头的操作只需要在 XY-MWC320A MQTT 发布者端执行即可，以 XY-MWC320A MQTT 订阅者开头的操作只需要在 XY-MWC320A MQTT 订阅者端执行即可。如果操作没有特别指明在哪端操作，则需要在发布者端和订阅者端都执行。

#### 1. 设置 MQTT 用户属性。

XY-MWC320A MQTT 发布者：

AT

```
1 AT+MQTTUSERCFG=0,1,"publisher","xyuser","123456789",0,0,""
```

Plain Text

```
1 OK
```

XY-MWC320A MQTT 订阅者：

AT

```
1 AT+MQTTUSERCFG=0,1,"subscriber","xyuser","123456789",0,0,""
```

Plain Text

```
1 OK
```

#### 2. 连接 MQTT 代理。

AT

```
1 AT+MQTTCONN=0,"192.168.3.102",8883,1
```

Plain Text

```
1 +MQTTCONNECTED:0,1,"192.168.3.102","8883","",1
2 OK
```

**说明：**输入的 MQTT 代理域名或 MQTT 代理 IP 地址可能与上述命令中的不同。

### 3. 订阅 MQTT 主题。

XY-MWC320A MQTT 订阅者：

AT

```
1 AT+MQTTSUB=0,"topic",1
```

Plain Text

```
1 OK
```

### 4. 发布 MQTT 消息（字符串）。

假设要发布消息的数据如下，长度为 427 字节。

AT

```
1 {"headers": {"Accept": "application/json","Accept-Encoding": "gzip, deflate","Accept-Language": "en-US,en;q=0.9,zh-CN;q=0.8,zh;q=0.7","Content-Length": "0","Host": "httpbin.org","Origin": "http://httpbin.org","Referer": "http://httpbin.org/","User-Agent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.114 Safari/537.36","X-Amzn-Trace-Id": "Root=1-6150581e-1ad4bd5254b4bf5218070413"}}}
```

XY-MWC320A MQTT 发布者：

AT

```
1 AT+MQTTPUBRAW=0,"topic",427,0,0
```

Plain Text

```
1 OK
2 >
```

上述响应表示 AT 已准备好接收串行数据，此时可以输入数据，当 AT 接收到的数据长度达到 <length> 后，数据传输开始。

Plain Text

```
1 +MQTTPUB:OK
```

AT

```
1 AT 输出 > 字符后，数据中的特殊字符不需要转义字符进行转义，也不需要以新行结尾（CR-LF）。
```

如果 XY-MWC320A MQTT 发布者成功发布消息，以下信息将会在 XY-MWC320A MQTT 订阅者端提示。

Plain Text

```
1 +MQTTSUBRECV:0,"topic",427,{"headers":{"Accept": "application/json","Accept-Encoding": "gzip, deflate","Accept-Language": "en-US,en;q=0.9,zh-CN;q=0.8,zh;q=0.7","Content-Length": "0","Host": "httpbin.org","Origin": "http://httpbin.org","Referer": "http://httpbin.org/","User-Agent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.114 Safari/537.36","X-Amzn-Trace-Id": "Root=1-6150581e-1ad4bd5254b4bf5218070413"}}
```

- AT

```
1 AT 输出 > 字符后，数据中的特殊字符不需要转义字符进行转义，也不需要以新行结尾（CR-LF）。
```

- 如果 XY-MWC320A MQTT 发布者成功发布消息，以下信息将会在 XY-MWC320A MQTT 订阅者端提示。

Plain Text

```
1 +MQTTSUBRECV:0,"topic",427,{"headers":{"Accept": "application/json","Accept-Encoding": "gzip, deflate","Accept-Language": "en-US,en;q=0.9,zh-CN;q=0.8,zh;q=0.7","Content-Length": "0","Host": "httpbin.org","Origin": "http://httpbin.org","Referer": "http://httpbin.org/","User-Agent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.114 Safari/537.36","X-Amzn-Trace-Id": "Root=1-6150581e-1ad4bd5254b4bf5218070413"}}
```

## 1. 关闭 MQTT 连接。

AT

```
1 AT+MQTTCLEAN=0
```

Plain Text

```
1 OK
```

### 2.5.3 基于 TLS 的 MQTT 连接（需要本地创建 MQTT 代理）

以下示例同时使用两块 XY-MWC320A 开发板，其中一块作为 MQTT 发布者（只作为 MQTT 发布者角色），另一块作为 MQTT 订阅者（只作为 MQTT 订阅者角色）。

示例介绍了如何基于 TLS 创建 MQTT 连接。首先需要创建一个本地 MQTT 代理，假设 MQTT 代理的 IP 地址为 192.168.3.102，端口为 8883。

#### 重要

步骤中以 XY-MWC320A MQTT 发布者 开头的操作只需要在 XY-MWC320A MQTT 发布者端执行即可，以 XY-MWC320A MQTT 订阅者 开头的操作只需要在 XY-MWC320A MQTT 订阅者端执行即可。如果操作没有特别指明在哪端操作，则需要在发布者端和订阅者端都执行。

#### 1. 设置时区和 SNTP 服务器。

AT

1 AT+CIPSNTPCFG=1,8,"ntp1.aliyun.com"

Plain Text

1 OK

## 2. 查询 SNTP 时间。

AT

1 AT+CIPSNTPTIME?

Plain Text

1 +CIPSNTPTIME:Thu Sep 2 18:57:03 2021

2 OK

**说明：**查询到的 SNTP 结果可能与上述响应中的不同。请确保 SNTP 时间一定是真实有效的时间，不能是 1970 年及之前的时间。设置时间是为了在 TLS 认证时校证书的有效期。

## 3. 设置 MQTT 用户属性。

XY-MWC320A MQTT 发布者：

AT

1 AT+MQTTUSERCFG=0,4,"publisher","xyuser","123456789",0,0,""

Plain Text

1 OK

XY-MWC320A MQTT 订阅者：

AT

1 AT+MQTTUSERCFG=0,4,"subscriber","xyuser","123456789",0,0,""

Plain Text

1 OK

## 4. 设置 MQTT 连接属性。

AT

1 AT+MQTTCONNCFG=0,0,0,"lwtt","lwtm",0,0

Plain Text

1 OK

## 5. 连接 MQTT 代理。

AT

1 AT+MQTTCONN=0,"192.168.3.102",8883,1

Plain Text

```
1 +MQTTCONNECTED:0,4,"192.168.3.102","8883","",1
2 OK
```

**说明：**输入的 MQTT 代理域名或 MQTT 代理 IP 地址可能与上述命令中的不同。

## 6. 订阅 MQTT 主题。

XY-MWC320A MQTT 订阅者：

AT

```
1 AT+MQTTSUB=0,"topic",1
```

Plain Text

```
1 OK
```

## 7. 发布 MQTT 消息（字符串）。

XY-MWC320A MQTT 发布者：

AT

```
1 AT+MQTTPUB=0,"topic","test",1,0
```

Plain Text

```
1 OK
```

**说明：**如果 XY-MWC320A MQTT 发布者成功发布消息，以下信息将会在 XY-MWC320A MQTT 订阅者端提示。

Plain Text

```
1 +MQTTSUBRECV:0,"topic",4,test
```

- 如果 XY-MWC320A MQTT 发布者成功发布消息，以下信息将会在 XY-MWC320A MQTT 订阅者端提示。

Plain Text

```
1 +MQTTSUBRECV:0,"topic",4,test
```

## 1. 关闭 MQTT 连接。

AT

```
1 AT+MQTTCLEAN=0
```

Plain Text

```
1 OK
```

### 2.5.4 基于 WSS 的 MQTT 连接

以下示例同时使用两块 XY-MWC320A 开发板，其中一块作为 MQTT 发布者（只作为 MQTT 发布者角色），另一块作为 MQTT 订阅者（只作为 MQTT 订阅者角色）。

示例介绍了如何基于 WSS 创建 MQTT 连接。MQTT 代理域名为 test.mosquitto.org，资源路径为 mqtt，端口为 8081。

## 重要

步骤中以 XY-MWC320A MQTT 发布者开头的操作只需要在 XY-MWC320A MQTT 发布者端执行即可，以 XY-MWC320A MQTT 订阅者开头的操作只需要在 XY-MWC320A MQTT 订阅者端执行即可。如果操作没有特别指明在哪端操作，则需要在发布者端和订阅者端都执行。

### 1. 设置时区和 SNTP 服务器。

AT

```
1 AT+CIPSNTPCFG=1,8,"ntp1.aliyun.com"
```

Plain Text

```
1 OK
```

### 2. 查询 SNTP 时间。

AT

```
1 AT+CIPSNTPTIME?
```

Plain Text

```
1 +CIPSNTPTIME:Thu Sep 2 18:57:03 2021
```

```
2 OK
```

**说明：**查询到的 SNTP 结果可能与上述响应中的不同。请确保 SNTP 时间一定是真实有效的时间，不能是 1970 年及之前的时间。设置时间是为了在 TLS 认证时校验证书的有效期。

### 3. 设置 MQTT 用户属性。

XY-MWC320A MQTT 发布者：

AT

```
1 AT+MQTTUSERCFG=0,7,"publisher","", "",0,0,"mqtt"
```

Plain Text

```
1 OK
```

XY-MWC320A MQTT 订阅者：

AT

```
1 AT+MQTTUSERCFG=0,7,"subscriber","", "",0,0,"mqtt"
```

Plain Text

```
1 OK
```

#### 4. 连接 MQTT 代理。

AT

```
1 AT+MQTTCONN=0,"test.mosquitto.org",8081,1
```

Plain Text

```
1 +MQTTCONNECTED:0,7,"test.mosquitto.org","8081","/mqtt",1
2 OK
```

**说明：** 输入的 MQTT 代理域名或 MQTT 代理 IP 地址可能与上述命令中的不同。

#### 5. 订阅 MQTT 主题。

XY-MWC320A MQTT 订阅者：

AT

```
1 AT+MQTTSUB=0,"topic",1
```

Plain Text

```
1 OK
```

#### 6. 发布 MQTT 消息（字符串）。

XY-MWC320A MQTT 发布者：

AT

```
1 AT+MQTTPUB=0,"topic","test",1,0
```

Plain Text

```
1 OK
```

**说明：** 如果 XY-MWC320A MQTT 发布者成功发布消息，以下信息将会在 XY-MWC320A MQTT 订阅者端提示。

Plain Text

```
1 +MQTTSUBRECV:0,"topic",4,test
```

- 如果 XY-MWC320A MQTT 发布者成功发布消息，以下信息将会在 XY-MWC320A MQTT 订阅者端提示。

Plain Text

```
1 +MQTTSUBRECV:0,"topic",4,test
```

#### 1. 关闭 MQTT 连接。

AT

```
1 AT+MQTTCLEAN=0
```

Plain Text

1 OK

## 2.6 MQTT AT 连接云示例

本文档主要介绍的设备如何通过 AT 命令对接 AWS IoT。

### 重要

有关如何使用 MQTT AT 命令的详细信息，请参阅 MQTT AT 命令集。

需要通过阅读 AWS IoT 开发指南 来熟悉 AWS IoT。

请按照以下步骤使用 XY-MWC320A 将的 XY-MWC320A 设备连接到 AWS IoT。

### 2.6.1 从 AWS IoT 获取证书以及 endpoint

1. 登录的 AWS IoT 控制台帐号，以及切换至 IoT Core services。
2. 按照 创建 AWS IoT 资源 中的说明创建 AWS IoT 策略、事物和证书。

确保已获得以下证书和密钥文件：

- device.pem.crt（设备证书）
- private.pem.key（私有密钥）
- Amazon-root-CA-1.pem（根 CA 证书）

1. 根据文档 设置策略 获取端点 (endpoint) 以及了解如何通过证书将事物绑定到策略。

端点的格式为 “xxx-ats.iot.us-east-2.amazonaws.com”。

### 备注

强烈建议熟悉 AWS IoT 开发人员指南。以下是本指南中值得注意的一些要点。

- AWS IoT 需要所有设备必须配备设备证书、私钥和根 CA 证书。
- 有关如何激活证书。
- 区域建议选择俄亥俄州 (Ohio)。

### 2.6.2 使用 MQTT AT 命令基于双向认证连接 AWS IoT

#### 2.6.2.1 配置证书

有三种方式可以替换 MQTT 证书：

方式一：重新编译固件

打开本地 XY-MWC320A 工程，并执行如下操作：

- 使用 Amazon-root-CA-1.pem 替换 mqtt\_ca.crt。

- 使用 device.pem.crt 替换 mqtt\_client.crt 。
- 使用 private.pem.key 替换 mqtt\_client.key 。

然后编译 XY-MWC320A 项目以构建 AT 固件，并将固件烧录到的 XY-MWC320A 设备。欲了解更多信息，请参阅 本地编译 XY-MWC320A 工程。

#### 方式二：运行时更新证书

如果不想重新编译固件，可以直接使用 AT+SYSMFG 命令在运行时更新 MQTT 证书。具体操作步骤请参考 AT+SYSMFG 命令示例中的 PKI 配置，证书配置方法与 SSL 证书相同，仅需将命名空间修改为 mqtt\_cert、mqtt\_key、mqtt\_ca 即可。

#### 方式三：仅更新证书 bin 文件

如果已有 AT 固件，只需预烧录自己的证书，可以直接更新 mfg\_nvs.bin 文件。具体操作步骤请参阅 如何更新 PKI 配置。

### 2.6.2.2 使用 AT 命令连接到 AWS IoT

#### 1. 设置 Wi-Fi 模式为 station。

```
AT
1 AT+CWMODE=1
```

```
Plain Text
1 OK
```

#### 2. 连接 AP。

```
AT
1 AT+CWJAP=<"ssid">,<"password">
```

```
Plain Text
1 OK
```

#### 3. 设置 SNTP Server。

```
AT
1 AT+CIPSNTPCFG=1,8,"pool.ntp.org"
```

```
Plain Text
1 OK
```

#### 4. 查询 SNTP 时间。

```
AT
1 AT+CIPSNTPTIME?
```

Plain Text

```
1 +CIPSNTPTIME:<asctime style time>
2 OK
```

**说明：**此时获得的 <asctime style time> 必须是设置时区的实时时间，否则会因为证书有效期而导致连接失败。

## 5. (可选) 运行时更新 MQTT 证书。

### a). 更新 MQTT 根 CA 证书。

AT

```
1 AT+SYSMFG=2,"mqtt_ca","mqtt_ca",8,<ca_len>
```

Plain Text

```
1 OK
2 >
```

收到 > 后，发送 Amazon-root-CA-1.pem 的完整内容，其中 <ca\_len> 为证书文件的字节数。

### b). 更新 MQTT 客户端私钥。

AT

```
1 AT+SYSMFG=2,"mqtt_key","mqtt_key",8,<key_len>
```

Plain Text

```
1 OK
2 >
```

收到 > 后，发送 private.pem.key 的完整内容，其中 <key\_len> 为私钥文件的字节数。

### c). 更新 MQTT 客户端证书。

AT

```
1 AT+SYSMFG=2,"mqtt_cert","mqtt_cert",8,<cert_len>
```

Plain Text

```
1 OK
2 >
```

收到 > 后，发送 device.pem.crt 的完整内容，其中 <cert\_len> 为证书文件的字节数。

**说明：**mqtt\_ca、mqtt\_key 和 mqtt\_cert 命名空间的键名分别与命名空间同名。写入的是二进制类型数据 (<type> = 8)，发送的数据长度必须与 <ca\_len>、<key\_len>、<cert\_len> 完全一致。如需查询或验证写入结果，可参考 AT+SYSMFG 命令示例。

## 6. 设置 MQTT 用户属性。

AT

```
1 AT+MQTTUSERCFG=0,5,"XY","XY-Router","1234567890",0,0,""
```

Plain Text

```
1 OK
```

AT

```
1 AT+MQTTUSERCFG 中第二参数为 5，即双向认证，不可更改。
```

- AT

```
1 AT+MQTTUSERCFG 中第二参数为 5，即双向认证，不可更改。
```

## 1. 连接 AWS IoT。

AT

```
1 AT+MQTTCONN=0,"<endpoint>",8883,1
```

Plain Text

```
1 +MQTTCONNECTED:0,5,<endpoint>,"8883","",1
2 OK
```

**说明：**请在 <endpoint> 参数处填入在 AWS IoT 控制台获取的实际 endpoint 字符串。无法更改端口 8883。

## 2. 订阅消息。

AT

```
1 AT+MQTTSUB=0,"topic/XYat",1
```

Plain Text

```
1 OK
```

## 3. 发布消息。

AT

```
1 AT+MQTTPUB=0,"topic/XYat","hello aws!",1,0
```

Plain Text

```
1 +MQTTSUBRECV:0,"topic/XYat",10,hello aws!
2 OK
```

### 2.6.2.3 示例日志

正常交互日志如下：

#### 1. XY-MWC320A 端日志

#### 2. AWS 端日志

## 2.7 HTTP AT 示例

本文档主要提供在 XY-MWC320A 设备上运行 HTTP AT 命令集 命令的详细示例。

### 重要

当前 XY-MWC320A 仅支持部分 HTTP 客户端的功能。

### 2.7.1 HTTP 客户端 HEAD 请求方法

该示例以 `http://httpbin.org` 作为 HTTP 服务器。

#### 1. 恢复出厂设置。

```
AT
1 AT+RESTORE
```

```
Plain Text
1 OK
```

#### 2. 设置 Wi-Fi 模式为 station。

```
AT
1 AT+CWMODE=1
```

```
Plain Text
1 OK
```

#### 3. 连接路由器。

```
AT
1 AT+CWLAP="XY-Router", "1234567890"
```

```
Plain Text
1 WIFI CONNECTED
2 WIFI GOT IP
3 OK
```

**说明：**实际输入的 SSID 和密码可能与上述命令中的不同，请使用所用路由器的 SSID 和密码。

#### 4. 发送一个 HTTP HEAD 请求。设置 opt 为 1（HEAD 方法），url 为 `http://httpbin.org/get`，transport\_type 为 1 (HTTP\_TRANSPORT\_OVER\_TCP)。

```
AT
1 AT+HTTPCLIENT=1,0,"http://httpbin.org/get",,,1
```

Plain Text

```
1 +HTTPCLIENT:35, Date: Sun, 26 Sep 2021 06:59:13 GMT
2 +HTTPCLIENT:30, Content-Type: application/json
3 +HTTPCLIENT:19, Content-Length: 329
4 +HTTPCLIENT:22, Connection: keep-alive
5 +HTTPCLIENT:23, Server: gunicorn/19.9.0
6 +HTTPCLIENT:30, Access-Control-Allow-Origin: *
7 +HTTPCLIENT:38, Access-Control-Allow-Credentials: true
8 OK
```

**说明：**获取到的 HTTP 头部信息可能与上述响应中的不同。

## 2.7.2 HTTP 客户端 GET 请求方法

本例以下载一个 JPG 格式的图片文件为例。图片链接为 <https://www.newbitinfo.com/dl/solution-platform.jpg>。

### 1. 恢复出厂设置。

AT

```
1 AT+RESTORE
```

Plain Text

```
1 OK
```

### 2. 设置 Wi-Fi 模式为 station。

AT

```
1 AT+CWMODE=1
```

Plain Text

```
1 OK
```

### 3. 连接路由器。

AT

```
1 AT+CWJAP="XY-Router", "1234567890"
```

Plain Text

```
1 WIFI CONNECTED
2 WIFI GOT IP
3 OK
```

**说明：**实际输入的 SSID 和密码可能与上述命令中的不同，请使用所用路由器的 SSID 和密码。

### 4. 发送一个 HTTP GET 请求。设置 opt 为 2（GET 方法），url 为 <https://www.newbitinfo.com/dl/solution-platform.jpg>，transport\_type 为 2

(HTTP\_TRANSPORT\_OVER\_SSL)。

AT

1 AT+HTTPCLIENT=2,0,"https://www.newbitinfo.com/dl/solution-platform.jpg",,,2

Plain Text

```

1 +HTTPCLIENT:512,<0xff><0xd8><0xff><0xe2><0x0c>XICC_PROFILE<break>
2 <0x01><0x01><break>
3 <break>
4 <0x0c>HLino<0x02><0x10><break>
5 <break>
6 mntrRGB XYZ <0x07><0xce><break>
7 <0x02><break>
8 ...
9 +HTTPCLIENT:512,<0xeb><0xe2>v<0xcb><0x98>-<0xf8><0x8a><0xae><0xf3><0xc8><0xb6><0
xa8><0x86><0x02>j<0x06><0xe2>
10 "<0xaa>*p<0x7f>2",h<0x12>N<0xa5><0x1e><0xd2>bp<0xea><0x1e><0xf5><0xa3>x<0xa6>J<
0x14>Ti<0xc3>m<0x1a>m<0x94>T<0xe1>I<0xb6><0x90><0xdc>_<0x11>QU;<0x94><0x97><0xc
b><0xdd><0xc7><0xc6><0x85><0xd7>U<0x02><0xc9>W<0xa4><0xaa><0xa1><0xa1><0x08>hB<
0x1a><0x10><0x86><0x84>!<0xa1><0x08>hB<0x1a><0x10><0x9b><0xb9>K<0xf5>5<0x95>5-=
<0x8a><0xcb><0xce><0xe0><0x91><0xf0>m<0xa9><0x04>C<0xde>k<0xe7><0xc2>v<H|<0xaf>
<0xb8>L<0x91>=<0xda>_<0x94><0xde><0xd0><0xa9><0xc0><0xdd>8<0x9a>B<0xaa><0x1a><0
x10><0x86><0x84>$<0xf4><0xd6><0xf2><0xa3><0x92><0xe7><0xa8>I<0xa3>b<0x1f>)<0xe1
>z<0xc4>y<0xae><0xca><0xed><0xec><0x1e>|^<0xd7>E<0xa2>_<0x13><0x9e>;{|<0xb5>Q<0
x97><0xa5>P<0xdf><0xa1>#3vn<0x1b><0xc3>-<0x92><0xe2>dIn<0x9c><0xb8>
11 <0xc7><0xa9><0xc6>(<0xe0><0xd3>i-<0x9e>@<0xbb><0xcc><0x88><0xd5>K<0xe3><0xf0>O<
0x9f>Km<0xb3>h<0xa8>omR<0xfe><0x8b><0xf9><0xa4><0xa6><0xff><break>
12 aU<0xdf><0xf3><0xa3>:A<0xe2>UG<0x04>k<0xaa>*<0xa1><0xa1><0x0b><0xca><0xec><0xd8
>Q<0xfb><0xbc>yqY<0xec><0xfb>?<0x16>CM<0xf6>|}<0xae><0xf3><0x1e><0xdf>%<0xf8><0
xe8><0xb1>B<0x8f>[<0xb3><0x04><0xec><0xeb>f<0x06><0x1c><0xe8><0x92><0xc9><0x8c
><0xb0>I<0xd1><0x8b>%<0x99><0x04><0xd0><0xbb>s<0x8b>xj<0xe2>4f<0xa0><0x8e>+E<0x
da><0xab><0xc7>=<0xab><0xc7><0xb9>xz1f<0xba><0xfd>_e6<0xff><break>
13 (w<0xa7>b<0xe3>m<0xf0>|<0x82><0xc9><0xfb><0x8b><0xac>r<0x95><0x94><0x96><0xd9>i
<0xe9>RVA<0x91><0x83><0x8b>M'<0x86><0x8f><0xa3>A<0xd8><0xd8>"r"<0x8a><0xa8><0x9
e>z1=<0xcd><0x16><0x07>D<0xa2><0xd0>u(<0xc2><0x8b><0x0b><0xc4><0xf1><0x87><0x9c
><0x93><0x8f><0xe3><0xd5>U<0x12>]<0x8e><0x91>]<0x91><0x06>#1<0xbe><0xf4>t0?<0xd
7><0x85><GEM<0xb1>%<0xee>UUT<0xe7><0xdf><0xa0><0xb9><0xce><0xe2>U@<0x03><0x82>S
<0xe9>*<0xa8>hB<0x1a><0x10><0xa1><0xaf>V<0x19>U<0x9d><0xb3>x<0xa6><0xc7><0xe2><
0x86><0x8e>d[<0x89>e<0x05>l<0x80>H<0x91>#<0xd2><0x8c><0xe1>j<0x1b>rH<0x04><0x89
><0x98><0xd3>lZW]q<0xc2><'>;<0x93><0xb4><0xf5>&<0x9d><0xa0>^Wp<0xa9>6`<0xe2>T<0
xa2><0xc2><0xb1>*<0xbc><0x13><0x13><0xa0><0xc4>)<0x83><0xb6><0xbe><0x86><0xb9><
0x88>-<0x1a>
14 OK
    
```

## 2.7.3 HTTP 客户端 POST 请求方法 (适用于 POST 少量数据)

该示例以 `http://httpbin.org` 作为 HTTP 服务器，数据类型为 `application/json`。

### 1. 恢复出厂设置。

```
AT
1 AT+RESTORE
```

```
Plain Text
1 OK
```

### 2. 设置 Wi-Fi 模式为 station。

```
AT
1 AT+CWMODE=1
```

```
Plain Text
1 OK
```

### 3. 连接路由器。

```
AT
1 AT+CWJAP="XY-Router", "1234567890"
```

```
Plain Text
1 WIFI CONNECTED
2 WIFI GOT IP
3 OK
```

**说明：**实际输入的 SSID 和密码可能与上述命令中的不同，请使用所用路由器的 SSID 和密码。

### 4. 发送一个 HTTP POST 请求。设置 opt 为 3（POST 方法），url 为 `http://httpbin.org/post`，content-type 为 1（`application/json`），transport\_type 为 1（`HTTP_TRANSPORT_OVER_TCP`）。

```
AT
1 AT+HTTPCLIENT=3,1,"http://httpbin.org/post",,,1,"{"form":{"purpose":"test\n"}}"
```

Plain Text

```
1 +HTTPCLIENT:282,{
2   "args": {},
3   "data": "{\"form\":{\"purpose\":\"test\"}}",
4   "files": {},
5   "form": {},
6   "headers": {
7     "Content-Length": "27",
8     "Content-Type": "application/json",
9     "Host": "httpbin.org",
10    "User-Agent": "XY HTTP Client/1.0",
11    "X-Amzn-Trace-Id": "Root=
12 +HTTPCLIENT:173,1-61503a3f-4b16b71918855b97614c5dfb"
13 },
14 "json": {
15   "form": {
16     "purpose": "test"
17   }
18 },
19 "origin": "20.187.154.207",
20 "url": "http://httpbin.org/post"
21 }
22 OK
```

**说明：**获取到的结果可能与上述响应中的不同。

## 2.7.4 HTTP 客户端 POST 请求方法（推荐方式）

如果 POST 的数据量相对较多，已经超过了单条 AT 命令的长度阈值 256，则建议可以使用 AT+HTTPCPOST 命令。

该示例以 http://httpbin.org 作为 HTTP 服务器，数据类型为 application/json。

### 1. 恢复出厂设置。

AT

```
1 AT+RESTORE
```

Plain Text

```
1 OK
```

### 2. 设置 Wi-Fi 模式为 station。

AT

```
1 AT+CWMODE=1
```

Plain Text

1 OK

### 3. 连接路由器。

AT

1 AT+CWJAP="XY-Router","1234567890"

Plain Text

1 WIFI CONNECTED  
2 WIFI GOT IP  
3 OK

**说明：**实际输入的 SSID 和密码可能与上述命令中的不同，请使用所用路由器的 SSID 和密码。

### 4. Post 指定长度数据。该命令设置 HTTP 头部字段数量为 2，分别是 connection 字段和 content-type 字段，connection 字段值为 keep-alive，content-type 字段值为 application/json。

假设想要 post 的 JSON 数据如下，长度为 427 字节。

AT

1 {"headers": {"Accept": "application/json", "Accept-Encoding": "gzip, deflate", "Accept-Language": "en-US,en;q=0.9,zh-CN;q=0.8,zh;q=0.7", "Content-Length": "0", "Host": "httpbin.org", "Origin": "http://httpbin.org", "Referer": "http://httpbin.org/", "User-Agent": "Mozilla/5.0 (X11; Linux x86\_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.114 Safari/537.36", "X-Amzn-Trace-Id": "Root=1-6150581e-1ad4bd5254b4bf5218070413"}}}

AT

1 AT+HTTPCPOST="http://httpbin.org/post",427,2,"connection: keep-alive","content-type: application/json"

Plain Text

1 OK  
2 >

上述响应表示 AT 已准备好接收串行数据，此时可以输入上面提到的 427 字节长的数据，当 AT 接收到的数据长度达到 <length> 后，数据传输开始。

Plain Text

1 +HTTPCPOST:281,{

"args": {},

AT

```
1  "data": "{\\"headers\\": {\\"Accept\\": \\"application/json\\",\\"Accept-Encoding\\":
  \\"gzip, deflate\\",\\"Accept-Language\\": \\"en-US,en;q=0.9,zh-CN;q=0.8,zh;q=0.7
  \\",\\"Content-Length\\": \\"0\\",\\"Host\\": \\"httpbin.org\\",\\"Origin\\": \\"http://http
  bin.org\\",\\"Referer\\": \\"htt
2  +HTTPCPOST:512,p://httpbin.org/\\",\\"User-Agent\\": \\"Mozilla/5.0 (X11; Linux x86_
  64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.114 Safari/537.36
  \\",\\"X-Amzn-Trace-Id\\": \\"Root=1-6150581e-1ad4bd5254b4bf5218070413\\"}}",
```

"files": {},

"form": {},

"headers": {

"Content-Length": "427",

"Content-Type": "application/json",

"Host": "httpbin.org",

"User-Agent": "XY HTTP Client/1.0",

AT

```
1  "X-Amzn-Trace-Id": "Root=1-61505e76-278b5c267aaf55842bd58b32"
```

},

"json": {

"headers": {

Plain Text

```
1  +HTTPCPOST:512,"Accept": "application/json",
```

"Accept-Encoding": "gzip, deflate",

AT

```
1  "Accept-Language": "en-US,en;q=0.9,zh-CN;q=0.8,zh;q=0.7",
```

"Content-Length": "0",

"Host": "httpbin.org",

"Origin": "http://httpbin.org",

"Referer": "http://httpbin.org/",

"User-Agent": "Mozilla/5.0 (X11; Linux x86\_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.114 Safari/537.36",

AT

```
1  "X-Amzn-Trace-Id": "Root=1-6150581e-1ad4bd5254b4bf5218070413"
```

```
}
},
"origin": "20.187.154"

Plain Text
1 +HTTPCPOST:45,.207",

"url": "http://httpbin.org/post"
}
```

```
Plain Text
1 SEND OK
```

```
AT
1 AT 输出 > 字符后, HTTP body 中的特殊字符不需要转义字符进行转义, 也不需要以新行结尾 (CR-LF)。
```

- AT
 

```
1 AT 输出 > 字符后, HTTP body 中的特殊字符不需要转义字符进行转义, 也不需要以新行结尾 (CR-LF)。
```

## 2.7.5 HTTP 客户端 PUT 请求方法 (适用于无数据情况)

该示例以 `http://httpbin.org` 作为 HTTP 服务器。PUT 请求支持 查询字符串参数 模式。

### 1. 恢复出厂设置。

```
AT
1 AT+RESTORE
```

```
Plain Text
1 OK
```

### 2. 设置 Wi-Fi 模式为 station。

```
AT
1 AT+CWMODE=1
```

```
Plain Text
1 OK
```

### 3. 连接路由器。

```
AT
1 AT+CWJAP="XY-Router", "1234567890"
```

Plain Text

```
1 WIFI CONNECTED
2 WIFI GOT IP
3 OK
```

**说明：**实际输入的 SSID 和密码可能与上述命令中的不同，请使用所用路由器的 SSID 和密码。

4. 发送一个 HTTP PUT 请求。设置 opt 为 4（PUT 方法），url 为 `http://httpbin.org/put`，transport\_type 为 1 (HTTP\_TRANSPORT\_OVER\_TCP)。

AT

```
1 AT+HTTPCLIENT=4,0,"http://httpbin.org/put?user=foo",,,1
```

Plain Text

```
1 +HTTPCLIENT:282,{
2   "args": {
3     "user": "foo"
4   },
5   "data": "",
6   "files": {},
7   "form": {},
8   "headers": {
9     "Content-Length": "0",
10    "Content-Type": "application/x-www-form-urlencoded",
11    "Host": "httpbin.org",
12    "User-Agent": "XY HTTP Client/1.0",
13    "X-Amzn-Trace-Id": "R
14 +HTTPCLIENT:140,oot=1-61503d41-1dd8cbe0056190f721ab1912"
15   },
16   "json": null,
17   "origin": "20.187.154.207",
18   "url": "http://httpbin.org/put?user=foo"
19 }
20 OK
```

**说明：**获取到的结果可能与上述响应中的不同。

## 2.7.6 HTTP 客户端 PUT 请求方法（推荐方式）

该示例以 `http://httpbin.org` 作为 HTTP 服务器，数据类型为 `application/json`。

1. 恢复出厂设置。

AT

```
1 AT+RESTORE
```

Plain Text

1 OK

## 2. 设置 Wi-Fi 模式为 station。

AT

1 AT+CWMODE=1

Plain Text

1 OK

## 3. 连接路由器。

AT

1 AT+CWJAP="XY-Router", "1234567890"

Plain Text

1 WIFI CONNECTED  
2 WIFI GOT IP  
3 OK

**说明：**实际输入的 SSID 和密码可能与上述命令中的不同，请使用所用路由器的 SSID 和密码。

## 4. PUT 指定长度数据。该命令设置 HTTP 头部字段数量为 2，分别是 connection 字段和 content-type 字段，connection 字段值为 keep-alive，content-type 字段值为 application/json。

假设想要 put 的 JSON 数据如下，长度为 427 字节。

AT

1 {"headers": {"Accept": "application/json", "Accept-Encoding": "gzip, deflate", "Accept-Language": "en-US,en;q=0.9,zh-CN;q=0.8,zh;q=0.7", "Content-Length": "0", "Host": "httpbin.org", "Origin": "http://httpbin.org", "Referer": "http://httpbin.org/", "User-Agent": "Mozilla/5.0 (X11; Linux x86\_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.114 Safari/537.36", "X-Amzn-Trace-Id": "Root=1-6150581e-1ad4bd5254b4bf5218070413"}}}

AT

1 AT+HTTPCPUT="http://httpbin.org/put",427,2,"connection: keep-alive","content-type: application/json"

Plain Text

1 OK  
2 >

上述响应表示 AT 已准备好接收串行数据，此时可以输入上面提到的 427 字节长的数据，当 AT 接收到的数据长度达到 <length> 后，数据传输开始。

Plain Text

```
1 +HTTPCPUT:281,{
```

"args": {},

AT

```
1  "data": "{\"headers\": {\"Accept\": \"application/json\", \"Accept-Encoding\":  
  \"gzip, deflate\", \"Accept-Language\": \"en-US,en;q=0.9,zh-CN;q=0.8,zh;q=0.7  
  \", \"Content-Length\": \"0\", \"Host\": \"httpbin.org\", \"Origin\": \"http://http  
  bin.org\", \"Referer\": \"htt  
2  +HTTPCPUT:512,p://httpbin.org/\", \"User-Agent\": \"Mozilla/5.0 (X11; Linux x86_6  
  4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.114 Safari/537.36  
  \", \"X-Amzn-Trace-Id\": \"Root=1-6150581e-1ad4bd5254b4bf5218070413\"}}\",
```

"files": {},

"form": {},

"headers": {

"Content-Length": "427",

"Content-Type": "application/json",

"Host": "httpbin.org",

"User-Agent": "XY HTTP Client/1.0",

AT

```
1  "X-Amzn-Trace-Id": "Root=1-635f7009-681be2d5478504dc5b83624a"
```

},

"json": {

"headers": {

Plain Text

```
1 +HTTPCPUT:512,"Accept": "application/json",
```

"Accept-Encoding": "gzip, deflate",

AT

```
1  "Accept-Language": "en-US,en;q=0.9,zh-CN;q=0.8,zh;q=0.7",
```

"Content-Length": "0",

"Host": "httpbin.org",

"Origin": "http://httpbin.org",

"Referer": "http://httpbin.org/",

"User-Agent": "Mozilla/5.0 (X11; Linux x86\_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.114 Safari/537.36",

AT

1 "X-Amzn-Trace-Id": "Root=1-6150581e-1ad4bd5254b4bf5218070413"

}

},

"origin": "52.246.135

Plain Text

1 +HTTPCPUT:43,.57",

"url": "http://httpbin.org/put"

}

Plain Text

1 SEND OK

AT

1 AT 输出 > 字符后, HTTP body 中的特殊字符不需要转义字符进行转义, 也不需要以新行结尾 (CR-LF)。

- AT  
1 AT 输出 > 字符后, HTTP body 中的特殊字符不需要转义字符进行转义, 也不需要以新行结尾 (CR-LF)。

## 2.7.7 HTTP 客户端 DELETE 请求方法

该示例以 http://httpbin.org 作为 HTTP 服务器。DELETE 方法用于删除服务器上的资源。DELETE 请求的实现依赖服务器。

### 1. 恢复出厂设置。

AT

1 AT+RESTORE

Plain Text

1 OK

### 2. 设置 Wi-Fi 模式为 station。

AT

1 AT+CWMODE=1

Plain Text

1 OK

### 3. 连接路由器。

AT

1 AT+CWJAP="XY-Router","1234567890"

Plain Text

1 WIFI CONNECTED  
2 WIFI GOT IP  
3 OK

**说明：**实际输入的 SSID 和密码可能与上述命令中的不同，请使用所用路由器的 SSID 和密码。

### 4. 发送一个 HTTP DELETE 请求。设置 opt to 5（DELETE 方法），url 为 `http://httpbin.org/delete`，transport\_type to 1 (HTTP\_TRANSPORT\_OVER\_TCP)。

AT

1 AT+HTTPCLIENT=5,0,"https://httpbin.org/delete",,,1

Plain Text

```
1 +HTTPCLIENT:282,{
2   "args": {},
3   "data": "",
4   "files": {},
5   "form": {},
6   "headers": {
7     "Content-Length": "0",
8     "Content-Type": "application/x-www-form-urlencoded",
9     "Host": "httpbin.org",
10    "User-Agent": "XY HTTP Client/1.0",
11    "X-Amzn-Trace-Id": "Root=1-61504289-468a41
12 +HTTPCLIENT:114,737b0d251672acec9d"
13 },
14 "json": null,
15 "origin": "20.187.154.207",
16 "url": "https://httpbin.org/delete"
17 }
18 OK
```

**说明：**获取到的结果可能与上述响应中的不同。

## 2.8 Sleep AT 示例

本文档简要介绍并举例说明如何在 XY-MWC320A 系列产品上使用 AT 命令设置睡眠模式。

## 2.8.1 简介

XY-MWC320A 系列采用先进的电源管理技术，可以在不同的电源模式之间切换。当前，XY-MWC320A 支持以下四种功耗模式（更多休眠模式请参考技术规格书）：

1. Active 模式：芯片射频处于工作状态。芯片可以接收、发射和侦听信号。
2. Modem-sleep 模式：CPU 可运行，时钟可被配置。Wi-Fi 基带、蓝牙基带和射频关闭。
3. Light-sleep 模式：CPU 暂停运行。RTC 存储器和外设以及 ULP 协处理器运行。任何唤醒事件（MAC、主机、RTC 定时器或外部中断）都会唤醒芯片。
4. Deep-sleep 模式：CPU 和大部分外设都会掉电，只有 RTC 存储器和 RTC 外设处于工作状态。

默认情况下，XY-MWC320A 会在系统复位后进入 Active 模式。当 CPU 不需要一直工作时，例如等待外部活动唤醒时，系统可以进入低功耗模式。

XY-MWC320A 的功耗，请参考 XY-MWC320A 系列芯片技术规格书。

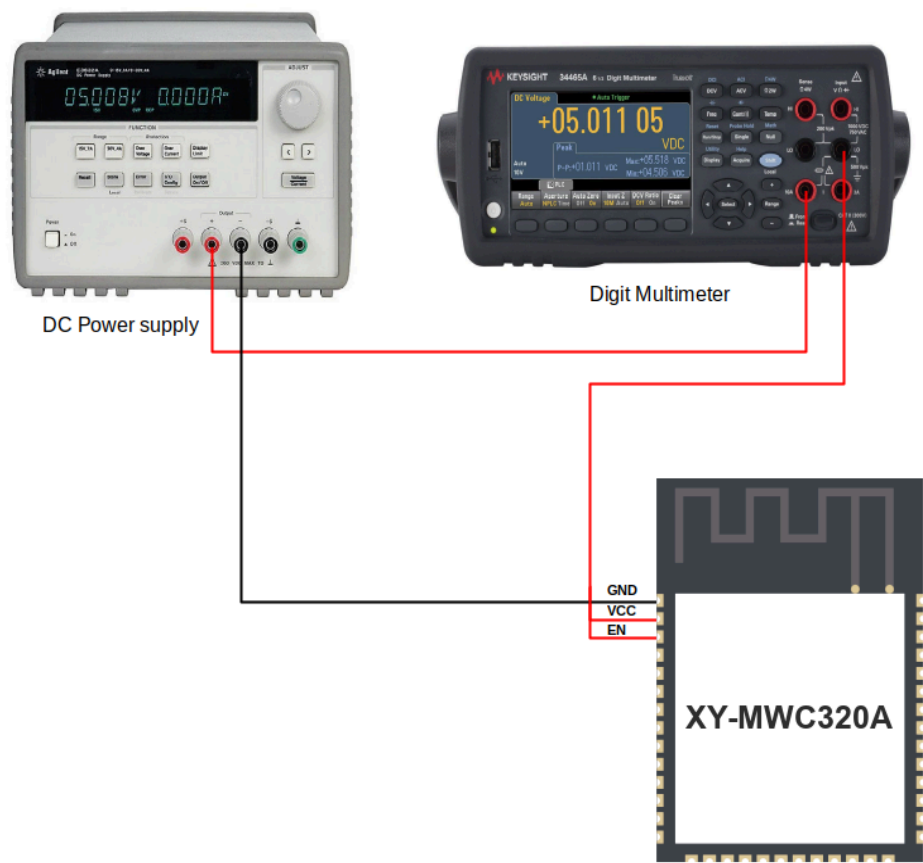
### 备注

- 将分别描述在 Wi-Fi 模式和蓝牙模式下将 XY-MWC320A 设置为睡眠模式。
- 在单 Wi-Fi 模式下，只有 station 模式支持 Modem-sleep 模式和 Light-sleep 模式。
- 在单蓝牙模式下的 Light-sleep 模式，如果配置了外部 32 kHz 晶振，功耗可降至  $\mu\text{A}$  级别。如果没有外部 32 kHz 晶振，系统将自动使用内部主晶振作为 RTC 时钟源，功耗约为几 mA。

### 2.8.1.1 测量方法

为避免功耗测试过程中出现一些不必要的干扰，建议使用集成芯片的XY 模组进行测试。

硬件连接可参考下图。（注意，图中开发板只保留了 XY-MWC320A，外围元器件均已移除。）



XY-MWC320A 硬件连接

2.8.2 在 Wi-Fi 模式下设置为 Modem-sleep 模式

1. 设置 Wi-Fi 为 station 模式。

```
AT
1 AT+CWMODE=1

Plain Text
1 OK
```

2. 连接路由器。

```
AT
1 AT+CWJAP="XY-Router", "1234567890"

Plain Text
1 WIFI CONNECTED
2 WIFI GOT IP
3 OK
```

说明：实际输入的 SSID 和密码可能与上述命令中的不同，请使用所用路由器的 SSID 和密码。

### 3. 设置休眠模式为 Modem-sleep 模式。

AT

1 AT+SLEEP=1

Plain Text

1 OK

#### 备注

RF 将根据 AP 的 DTIM 定期关闭（路由器一般设置 DTIM 为 1）。

## 2.8.3 在 Wi-Fi 模式下设置为 Light-sleep 模式

### 1. 设置 Wi-Fi 为 station 模式。

AT

1 AT+CWMODE=1

Plain Text

1 OK

### 2. 连接路由器。设置监听间隔为 3。

AT

1 AT+CWJAP="XY-Router","1234567890",,,3

Plain Text

1 WIFI CONNECTED

2 WIFI GOT IP

3 OK

**说明：**实际输入的 SSID 和密码可能与上述命令中的不同，请使用所用路由器的 SSID 和密码。

### 3. 设置休眠模式为 Light-sleep 模式。

AT

1 AT+SLEEP=2

Plain Text

1 OK

#### 备注

CPU 将会自动休眠，RF 则会根据 AT+CWJAP 设置的监听间隔定期关闭。

## 2.8.4 在蓝牙广播态下设置为 Light-sleep 模式

### 1. 初始化为角色为蓝牙服务端。

AT

1 AT+BLEINIT=2

Plain Text

1 OK

### 1. 设置蓝牙广播参数。设置蓝牙广播间隔为 1 s。

AT

1 AT+BLEADVPARAM=1600,1600,0,0,7,0,0,"00:00:00:00:00:00"

Plain Text

1 OK

### 1. 开始广播。

AT

1 AT+BLEADVSTART

Plain Text

1 OK

### 1. 禁用 Wi-Fi。

AT

1 AT+CWINIT=0

Plain Text

1 OK

### 1. 设置休眠模式为 Light-sleep 模式。

AT

1 AT+SLEEP=2

Plain Text

1 OK

## 2.8.5 在蓝牙连接态下设置为 Light-sleep 模式

### 1. 初始化为角色为蓝牙服务端。

AT

1 AT+BLEINIT=2

Plain Text

1 OK

## 1. 开始广播。

AT

1 AT+BLEADVSTART

Plain Text

1 OK

## 1. 等待连接。

如果连接建立成功，则 AT 将会提示：

Plain Text

```
1 +BLECONN:0,"47:3f:86:dc:e4:7d"
2 +BLECONNPARAM:0,0,0,6,0,500
3 +BLECONNPARAM:0,0,0,24,0,500
4 OK
```

- 在这个示例中，蓝牙客户端的地址为 47:3f:86:dc:e4:7d。
- 对于提示信息（+BLECONN and +BLECONNPARAM），请参考 AT+BLECONN 和 AT+BLECONNPARAM 获取更多信息。

## 1. 更新蓝牙连接参数。设置蓝牙连接间隔为 1 s。

AT

1 AT+BLECONNPARAM=0,800,800,0,500

Plain Text

1 OK

如果连接参数更新成功，则 AT 将会提示：

Plain Text

```
1 +BLECONNPARAM:0,800,800,800,0,500
```

- 对于提示信息（+BLECONNPARAM），请参考 AT+BLECONNPARAM 获取更多信息。

## 1. 禁用 Wi-Fi。

AT

1 AT+CWINIT=0

Plain Text

1 OK

1. 设置休眠模式为 Light-sleep 模式。

AT

1 AT+SLEEP=2

Plain Text

1 OK

2.8.6 设置为 Deep-sleep 模式

设置休眠模式为 Deep-sleep 模式。设置 deep-sleep 时间为 3600000 ms。

AT

1 AT+GSLP=3600000

Plain Text

1 OK

- 设定时间到后，设备自动唤醒，调用深度睡眠唤醒桩，然后加载应用程序。
- 对于 Deep-sleep 模式，唯一的唤醒方法是定时唤醒。

2.9 AT 配网示例

本文档主要介绍以下几种 XY-MWC320A 支持的配网方式：

- BluFi 配网
- SmartConfig 配网
- SoftAP 配网（WEB 配网）
- WPS 配网

| 配网方式          | BluFi 配网 | SmartConfig 配网 | SoftAP 配网 | WPS 配网 |
|---------------|----------|----------------|-----------|--------|
| 默认固件支持情况      |          |                | （ 备注 1 ）  |        |
| 是否需要 BLE      |          |                |           |        |
| 是否需要额外的手机 APP |          | （ 备注 2 ）       |           |        |
| 配网成功率         | 高        | 较高             | 高         | 高      |
| 操作复杂性         | 简单       | 简单             | 复杂        | 复杂     |
| 推荐指数          | 推荐       | 中等             | 中等        | 中等     |

- 备注 1：若想 AT 固件中支持 SoftAP 配网，请参考： Web 服务器 AT 命令。

- 备注 2：使用 SmartConfig 中的 AirKiss 类型配网时需依赖微信 APP，国外用户可能无法使用微信 APP。

### 2.9.1 BluFi 配网

BluFi 是一款基于蓝牙通道的 Wi-Fi 网络配置功能，更多信息请参考 BluFi 文档。

#### 1. 在手机端安装 EspBluFi 应用程序。

Android：安卓端 EspBluFi 下载（安卓端源码）

iOS：iOS 端 EspBluFi 下载（iOS 端源码）

#### 2. XY-MWC320A 设备设置 BluFi 名称。

```
AT
1 AT+BLUFINAME="blufi_test"
```

```
Plain Text
1 OK
```

#### 3. 开启 BluFi。

```
AT
1 AT+BLUFI=1
```

```
Plain Text
1 OK
```

#### 4. 手机创建 BluFi 连接并进行配网。

##### 4.1 BluFi 连接

手机打开系统蓝牙和 GPS 定位功能，启动 EspBluFi 应用程序，找到名为 blufi\_test 的设备并点击进入，再点击 连接 进行连接。此时，XY-MWC320A 设备端应打印类似于 +BLUFICONN 的日志，应用程序页面也会输出类似以下信息，表明 BluFi 连接已成功建立。

Connected <mac>

Discover service and characteristics success

Set notification enable complete

```
AT
1 Set mtu complete, mtu=...
```

##### 4.2 BluFi 配网

连接完成后，在 EspBluFi 应用程序中点击 配网 按钮，页面将会跳转至 配置 页面。在此页面输入 Wi-Fi 的 SSID 和密码，并点击 确定 按钮开始配网。稍等片刻，应用程序页面也会显示类似以下信息。

Post configure params complete

Receive device status response:

OpMode: Station

Station connect Wi-Fi now

Station connect Wi-Fi bssid: <mac>

Station connect Wi-Fi ssid: <ssid>

### 4.3 配网成功

请稍等片刻，XY-MWC320A 设备将输出以下日志。至此，XY-MWC320A 设备的配网已成功完成。

Plain Text

```
1 WIFI CONNECTED
2 WIFI GOT IP
```

### 5. (可选步骤) XY-MWC320A 设备发送 BluFi 用户自定义数据。

AT

```
1 AT+BLUFISEND=4
```

Plain Text

```
1 >
```

输入 4 字节数据，例如输入数据是 test，之后 AT 将会输出以下信息。

Plain Text

```
1 OK
```

此时手机应用程序端显示了类似以下信息：

Receive custom data:

test

### 6. XY-MWC320A 设备关闭 BluFi。

AT

```
1 AT+BLUFI=0
```

Plain Text

```
1 +BLUFIDISCONN
2 OK
```

此时手机应用程序端显示了类似以下信息：

AT

```
1 Disconnect <mac>, status=19
```

## 2.9.2 SmartConfig 配网

AT

- 1 AT SmartConfig 配网有四种类型，分别是 ESP-TOUCH、AirKiss、ESP-TOUCH+AirKiss 和 ESP-TOUCH v2。以下是 AirKiss 和 ESP-TOUCH v2 类型的配网示例。

### 2.9.2.1 ESP-TOUCH v2 配网示例

1. 在手机端安装 EspTouch 应用程序。

Android：安卓端 EspTouch 下载（安卓端应用程序源码）

iOS：iOS 端 EspTouch 下载（iOS 端应用程序源码）

2. 设置 XY-MWC320A 设备为 Station 模式。

AT

- 1 AT+CWMODE=1

Plain Text

- 1 OK

3. XY-MWC320A 设备开启 SmartConfig。

AT

- 1 AT+CWSTARTSMART=4,0,"1234567890123456"

Plain Text

- 1 OK

4. 在手机端进行配网。

请按照以下步骤操作：打开手机的 Wi-Fi，连接到目标网络（例如 SSID：test，密码：1234567890），启动 EspTouch 应用程序，点击 EspTouch V2 按钮。在跳转的页面中输入所连接 Wi-Fi 的密码、待配网的设备数量以及 AES 密钥，如下所示：

Wi-Fi 密码：1234567890

需要配网的设备数量：1

AES 密钥：1234567890123456

5. 在 XY-MWC320A 设备端确认配网成功。

此时 XY-MWC320A 设备端会输出类似如下信息：

Plain Text

- 1 smartconfig type:ESPTOUCH\_V2

Smart get wifi info

ssid:test

password:1234567890

Plain Text

```
1 WIFI CONNECTED
2 WIFI GOT IP
3 smartconfig connected wifi
```

到此，XY-MWC320A 设备的配网已成功完成。

## 6. XY-MWC320A 设备停止 SmartConfig。

AT

```
1 AT+CWSTOPSMART
```

Plain Text

```
1 OK
```

### 2.9.2.2 AirKiss 配网示例

1. 打开手机微信，搜索并关注支持 AirKiss 配网的公众号。

2. 设置 XY-MWC320A 设备为 Station 模式。

AT

```
1 AT+CWMODE=1
```

Plain Text

```
1 OK
```

3. XY-MWC320A 设备开启 SmartConfig。

AT

```
1 AT+CWSTARTSMART=2
```

Plain Text

```
1 OK
```

4. 在手机端进行配网。

手机连接到要配网的 Wi-Fi（例如 SSID: test，密码：1234567890），然后打开微信，进入支持 AirKiss 配网的公众号，点击产品资源，找到 AirKiss 设备并进入。在跳转的页面中输入 Wi-Fi 密码 1234567890，稍等片刻，手机页面将显示类似日志 配置成功.....。

此时 XY-MWC320A 设备端会输出类似如下信息：

Plain Text

```
1 smartconfig type:AIRKISS
```

Smart get wifi info

ssid:test

password:1234567890

Plain Text

```
1 WIFI CONNECTED
2 WIFI GOT IP
3 smartconfig connected wifi
```

到此，XY-MWC320A 设备的配网已成功完成。

## 5. XY-MWC320A 设备停止 SmartConfig。

AT

```
1 AT+CWSTOPOSMART
```

Plain Text

```
1 OK
```

### 2.9.3 SoftAP 配网

AT

```
1 AT SoftAP 配网指的是 WEB 配网。有关详细信息，请参见 Web Server AT 示例。
```

### 2.9.4 WPS 配网

1. 准备一个支持 WPS 配网的路由器（例如 Wi-Fi SSID: test）。
2. 设置 XY-MWC320A 设备为 Station 模式。

AT

```
1 AT+CWMODE=1
```

Plain Text

```
1 OK
```

## 3. XY-MWC320A 设备开启 WPS 配网。

AT

```
1 AT+WPS=1
```

Plain Text

```
1 OK
```

此时根据路由器的使用手册，开启 WPS 功能，稍等片刻，XY-MWC320A 设备端会输出类似如下信息，即设备配网已成功。

Plain Text

- 1 WIFI CONNECTED
- 2 WIFI GOT IP

#### 4. 查询 XY-MWC320A 设备连接上的 Wi-Fi 信息。

AT

- 1 AT+CWJAP?

Plain Text

- 1 +CWJAP:"test",.....
- 2 OK

#### XY-MWC320A 设备关闭 WPS 配网。

AT

- 1 AT+WPS=0

Plain Text

- 1 OK