GigaDevice Semiconductor Inc.

GD32VW553 AT 指令用户指南

应用笔记

AN151

1.5 版本

（2026 年 4 月）

目录

表索引

1. AT 指令格式

1.1. 指令类型

表 1-1. 指令类型

类型	格式	描述
帮助指令	AT+<x>=?	查看指令参数及取值范围
查询指令	AT+<x>?	查询指定目标的当前参数值
执行指令	AT+<x> 或 AT+<x>=<…>	运行命令 设置指定目标参数值

1.2. 指令格式

表 1-2. 指令格式

字段	说明
AT	指令前缀
<CMD>	指令字符串
[ ]	可选部分
<>	强制部分，针对特定命令，有些参数是强制要输入的
[p1],[p2],[p3],…	参数，参数支持字符串和数字两种，IP 地址采用字符串“x. x. x. x”格式输入 字符串：必须用双引号括起来 数字：支持十进制和十六进制
<CR LF>	换行

Note： AT [+<CMD>] [=] [p1],[p2],[p3],…<CR LF>

当支持 AT MQTT 指令集时，每条 AT 命令的总长度不能超过 256 字节。

1.3. 响应格式

表 1-3. 响应格式

输出类型	说明
[+<CMD>:<MSG>]	输出结果或错误提示
<RSP>	OK：代表成功 ERROR：代表失败

注意： 响应格式里面的汉字仅仅是对命令响应的解释，实际上不会显示。

2. AT 指令一览表

表 2-1. AT 指令

指令	描述
AT	测试 AT 指令模式
AT+HELP	查询所有 AT 指令
AT+RST	模块复位
AT+GMR	查询版本信息
AT+TASK	查询当前操作系统所有任务
AT+HEAP	查询当前操作系统空余 HEAP
AT+SYSRAM	查询当前空余 SRAM 空间
AT+UART	设置 LOG UART 参数或读取当前参数
AT+FS	操作文件系统
AT+FSMOUNT	挂载/卸载文件系统
AT+CWMODE_CUR	查询或设置 WiFi 当前工作模式：SoftAP/STA/MONITOR
AT+CWJAP_CUR	连接 AP
AT+CWLAP	扫描并显示 AP 列表
AT+CWSTATUS	查询 WiFi 当前工作模式和状态
AT+CWQAP	断开与 AP 的连接
AT+CWSAP_CUR	启动 SoftAP 模式
AT+CWLIF	查询所有连接到 SoftAP 的 STA 信息
AT+CWAUTOCONN	设置上电时是否自动连接 AP
AT+CIPAP	查询或设置 SoftAP 的 IP 地址
AT+PING	Ping 功能
AT+CIPSTA	查询或设置本地 STA 的 IP 地址
AT+CIPSTART	建立 TCP 连接或 UDP 传输
AT+CIPSEND	发送数据
AT+CIPSERVER	建立 TCP/UDP 服务器
AT+CIPCLOSE	关闭 TCP 连接或 UDP 传输
AT+CIPSTATUS	查询 Wi-Fi 连接信息
AT+CIPMUX	查询或启用/禁用多连接模式
AT+CIPSTATE	查询 TCP/UDP 连接信息
AT+CIPMODE	查询或设置传输模式
AT+TRANSINTVL	查询或设置透传模式下的数据发送间隔
AT+CIFSR	查询本地 IP 地址信息
AT+CIPDOMAIN	域名解析
AT+CIPSNTPCFG	查询或设置时区和 SNTP 服务器
AT+CIPSNTPTIME	查询 SNTP 时间
AT+CIPSNTPINTV	查询或设置 SNTP 时间同步间隔
AT+WSCFG	设置 WebSocket 参数
AT+WSHEAD	查询或设置 WebSocket 请求头

指令	描述
AT+WSOPEN	查询或打开一个 WebSocket 连接
AT+WSSEND	向 WebSocket 连接发送数据
AT+WSCLOSE	关闭 WebSocket 连接
AT+MQTTUSERCFG	设置 MQTT 用户属性
AT+MQTTLONGCLIENTID	设置 MQTT 客户端 ID
AT+MQTTLONGUSERNAME	设置 MQTT 登录用户名
AT+MQTTLONGPASSWORD	设置 MQTT 登录密码
AT+MQTTCONNCFG	设置 MQTT 连接属性
AT+MQTTCONN	连接 MQTT Broker 或查询 MQTT 状态
AT+MQTTPUB	发布 MQTT 消息（字符串）
AT+MQTTPUBRAW	发布长 MQTT 消息
AT+MQTTSUB	订阅 MQTT Topic 或查询已订阅 Topic
AT+MQTTUNSUB	取消订阅 MQTT Topic
AT+MQTTCLEAN	断开 MQTT 连接并释放资源
AT+HTTPCLIENT	发送 HTTP 客户端请求
AT+HTTPGETSIZE	获取 HTTP 资源大小
AT+HTTPCGET	获取 HTTP 资源
AT+HTTPCPOST	Post 指定长度的 HTTP 数据
AT+HTTPCPUT	Put 指定长度的 HTTP 数据
AT+HTTPURLCFG	查询或设置长的 HTTP URL
AT+HTTPCHEAD	查询或设置 HTTP 请求头
AT+BLEENABLE	使能 ble 模块
AT+BLEDISABLE	失能 ble 模块
AT+BLENAME	设置名称
AT+BLEADVSTART	开启蓝牙广播
AT+BLEADVSTOP	停止蓝牙广播
AT+BLEADVDATA	设置广播内容
AT+BLEADVDATAEX	设置广播内容
AT+BLESCANRSPDATA	设置扫描回复内容
AT+BLEPASSTH	SERVER 开启透传模式
AT+BLEPASSTHAUTO	自动开启透传模式
AT+BLEPASSCLI	CLIENT 开启透传模式
AT+BLESCANPARAM	设置扫描参数
AT+BLESCAN	开启扫描
AT+BLECONN	BLE 建立连接
AT+BLESCONNPARAM	设置/查询连接参数
AT+BLEDISCONN	BLE 断开连接
AT+BLEMTU	更新/查询 mtu
AT+BLEPHY	更新/查询 phy
AT+BLEDATALEN	Data length extension
AT+BLEADDR	查询/设置 ble bd address

指令	描述
AT+BLESETAUTH	配置 AUTHENTICATION
AT+BLEPAIR	发起配对
AT+BLEENCRYPT	启动加密
AT+BLEPASSKEY	输入 passkey
AT+BLECOMPARE	输入 compare 结果
AT+BLELISTENCDEV	列出 bond device 列表
AT+BLECLEARENCDEV	移除 bond 设备
AT+BLEGATTSSVC	列出本地注册的 service
AT+BLEGATTSLISTALL	列出本地所有 service 中信息
AT+BLEGATTSNTF	发送 notification
AT+BLEGATTSIND	发送 indication
AT+BLEGATTSSETATTRVAL	设置 characteristic 的值
AT+BLEGATTCDISCSVC	发现 service
AT+BLEGATTCDISCCHAR	发现 characteristic
AT+BLEGATTCDISCDESC	发现 descriptor
AT+BLEGATTCRD	读 attribute value
AT+BLEGATTCWR	写 attribute value
AT+ BLEDADATRANS	开启关闭普通透传模式
AT+BLEDADATRANSSEND	普通透传模式发送数据
AT+BLECOURIER	开启关闭配网模式

3. AT 基础指令集

3.1. AT

表 3-1. 测试 AT 指令模式

指令	参数	响应
执行指令 AT		执行结果
示例： AT 正确响应： OK		

3.2. AT+HELP

表 3-2. 查询所有 AT 指令

指令	参数	响应
执行指令 AT+HELP		显示全部 AT 命令列表
示例： AT+HELP 正确响应： AT COMMAND LIST: ========== AT AT+HELP …… OK		

3.3. AT+RST

表 3-3. 模块复位指令

指令	参数	响应
执行指令 AT+RST		重启消息
示例： AT+RST 正确响应： OK ==========		

SDK Version: v1.0.3-908337de9f3c0871 Build date: 2025/07/28 16:15:20 OK READY

3.4. AT+GMR

表 3-4. 查询版本信息

指令	参数	响应(类似格式信息)
执行指令 AT+GMR		相关版本信息
示例： AT+GMR 正确响应： ========== SDK Version: v1.0.3-908337de9f3c0871 Build date: 2025/07/28 16:15:20 OK		

3.5. AT+TASK

表 3-5. 查询当前操作系统所有任务

指令	参数	响应(类似格式信息)
执行指令 AT+TASK		当前 task 信息列表
示例： AT+TASK 正确响应： ATCMD X 20 402 2 0x2001a780 ... RX B 18 416 6 0x200203c8 OK		

3.6. AT+HEAP

表 3-6. 查询当前操作系统空余 HEAP

指令	参数	响应(类似格式信息)
执行指令 AT+HEAP		heap 使用情况
示例： AT+HEAP		

正确响应： ========== Total free heap size = 113784 Total min free heap size = 109480 OK

3.7. AT+SYSRAM

表 3-7. 查询当前空余 SRAM 空间

指令	参数	响应(类似格式信息)
执行指令 AT+SYSRAM		剩余 SRAM 空间
示例： AT+SYSRAM 正确响应： ========== Free SRAM size = 108472 OK		

3.8. AT+UART

表 3-8. 查询或设置串口参数

指令	参数	响应
帮助指令 AT+UART=?		+UART=<baudrate>,<databits>,<stopbits>,<parity>,<flow control>
查询指令 AT+UART?		当前串口参数
执行指令 AT+UART=<baudrate>,<databits>,<stopbits>,<parity>,<flow control>	<baudrate>：UART 波特率 <databits>：数据位 8：8 bit <stopbits>：停止位 1：1 bit 2：1.5 bit 3：2 bit <parity>：校验位 0：None 1：Odd 2：Even <flow control>：流控 0：不使能流控 1：使能 RTS 2：使能 CTS	执行结果

指令	参数	响应
	3：同时使能 RTS 和 CTS	
示例： AT+UART=115200,8,1,0,0 正确响应： OK		

3.9. AT+TRANSINTVL

表 3-9. 设置或查询透传模式下的数据发送间隔

指令	参数	响应
帮助指令 AT+TRANSINTVL=?		+TRANSINTVL=<interval>
查询指令 AT+TRANSINTVL?		当前数据发送间隔 +TRANSINTVL:<interval>
执行指令 AT+TRANSINTVL =<interval>	<interval>：数据发送间隔。整型，单位：毫秒，范围：0~1000，默认值 20	执行结果
示例： AT+TRANSINTVL=800 正确响应： OK		

3.10. AT+FS

表 3-10. 操作文件系统

指令	参数	响应
帮助指令 AT+FS=?		+FS=<type>,<operation>,<filename>,<offset>,<length>
执行指令 AT+FS=<type>,<operation>,<filename>,<offset>,<length>	<type>：文件系统类型，仅支持 FatFS，整型 0：FatFS <operation>：对文件系统的操作 0：删除文件或删除文件夹及文件夹内所有文件 1：写文件 2：读文件 3：查询文件大小 4：查询路径下文件 <filename>：文件名称，字符串参数 <offset>：偏移地址，仅针对读	执行结果 若<operation>为 1： > <input from keyboad> OK 若<operation>不为 1： OK

指令	参数	响应
	写操作有效，整型 <length>：长度，仅针对读写操作有效，整型，单位：字节	

示例 1：
AT+FS=0,1,"a/e.txt",0,10
正确响应 1：
>
OK

示例 2：
AT+FS=0,2,"a/e.txt",0,5
正确响应 2：
FATFS: read 5Bytes, content:
12345
OK

示例 3：
AT+FS=0,3,"a/e.txt"
正确响应 3：
10Bytes
OK

示例 4，查询根目录下所有文件：
AT+FS=0,4,"."
正确响应 4：
DIR: //a
FILE: //a/e.txt 10Bytes
OK

示例 5：
AT+FS=0,0,"a"
正确响应 5：
OK

注：
本命令会自动挂载文件系统，使用本命令完成文件系统的操作后，强烈建议使用 AT+FSMOUNT=0 命令卸载文件系统以释放内存空间。
当读取数据的长度大于实际文件的大小时，仅返回实际长度的数据。

3.11. AT+FSMOUNT

表 3-11. 挂载/卸载文件系统

指令	参数	响应
帮助指令 AT+FSMOUNT=?		+FSMOUNT=<mount>
执行指令 AT+FSMOUNT=<mount>	<mount>：挂载或卸载文件系统，整型 0：卸载文件系统 1：挂载文件系统	执行结果
示例 1： AT+FSMOUNT=1 正确响应 1： OK 示例 2： AT+FSMOUNT=0 正确响应 2： OK 注： 使用 AT+FS 命令完成文件系统的操作后，强烈建议使用本命令 AT+FSMOUNT=0 卸载文件系统以释放内存空间。		

4. AT WIFI 指令集

4.1. AT+CWMODE_CUR

表 4-1. 查询或设置 Wi-Fi 当前工作模式

指令	参数	响应
帮助指令 AT+CWMODE_CUR=?		+CWMODE_CUR=<mode:0-3>
查询指令 AT+CWMODE_CUR?		当前工作模式 +CWMODE_CUR:<mode>
执行指令 AT+CWMODE_CUR=<mode>	<mode>： 0：MONITOR 模式 1：STA 模式 2：SoftAP 模式 3：Wi-Fi 共存模式	执行结果
示例： AT+CWMODE_CUR=2 正确响应： OK		

4.2. AT+CWJAP_CUR

表 4-2. 查询已连接 AP 信息或连接 AP

指令	参数	响应
帮助指令 AT+CWJAP_CUR=?		+CWJAP_CUR=<ssid>,<pwd>
查询指令 AT+CWJAP_CUR?		+CWJAP_CUR: <ssid>,<mac>,<channel>,<rssi>
执行指令 AT+CWJAP_CUR=<ssid>, <pwd>	<ssid>：字符串参数 <pwd>：字符串参数	执行结果
示例 1： AT+CWJAP_CUR="totolink","12345678" 正确响应 1： WIFI CONNECTED OK 示例 2： AT+CWJAP_CUR="tplink","" 正确响应 2： WIFI CONNECTED		

指令	参数	响应
OK		

4.3. AT+CWLAP

表 4-3. 扫描并列出周围 AP 的信息

指令	参数	响应
帮助指令 AT+CWLAP=?		+CWLAP=[ssid]
执行指令 AT+CWLAP[=<ssid>]	<ssid>：字符串参数	扫描结果 +CWLAP: <ssid>,<rssi>,<mac>,<channel>,<encrypt>
示例 1： AT+CWLAP 正确响应 1： +CWLAP: iQOO Neo5, -44, d6:4f:86:cb:c8:d0, 1, WPA2 CCMP; +CWLAP: GD-guest, -43, 08:3a:38:cc:2f:d1, 1, OPEN; +CWLAP: OpenWrt, -33, c4:70:ab:d9:bd:11, 1, OPEN; +CWLAP: GD-internet, -44, 08:3a:38:cc:2f:d0, 1, OPEN; +CWLAP: Redmi K40, -56, ba:fa:07:50:63:f6, 1, WPA2 CCMP; +CWLAP: D-Link_DIR-822, -30, 1c:5f:2b:fd:be:60, 1, WPA2 CCMP; +CWLAP: iPhone 24 Pro Max Ultr, -48, fa:da:47:72:f0:b3, 2, WPA2 CCMP; +CWLAP: TP-LINK_8659, -20, 68:77:24:bd:86:59, 4, WPA2/WPA3 CCMP; OK 示例 2，如果带参数 ssid，就只显示相应 AP 信息： AT+CWLAP= "xiaomi_4a" 正确响应 2： +CWLAP: xiaomi_4a, -55, 88:c3:97:0d:c3:70, 1, OPEN; OK		

4.4. AT+CWSTATUS

表 4-4. 查询 WiFi 状态，STA 或者 SoftAP 或者 MONITOR

指令	参数	响应
执行指令 AT+CWSTATUS		+CWSTATUS: STA, connected, <ssid>,<channel>,<mac> 或 +CWSTATUS: MONITOR, <channel>, <mac>

指令	参数	响应
		或 +CWSTATUS: STA, disconnected 或 +CWSTATUS: SoftAP, <ssid>, <password>, <channel>
示例： AT+CWSTATUS 正确响应： +CWSTATUS: STA, connected, xiaomi_4a, 1, 76:ba:ed:20:22:a2 OK		

4.5. AT+CWQAP

表 4-5. 断开 AP

指令	参数	响应
执行指令 AT+CWQAP		断开连接消息
示例： AT+CWQAP 正确响应： OK		

4.6. AT+CWSAP_CUR

表 4-6. 启动 SoftAP

指令	参数	响应
帮助指令 AT+CWSAP_CUR=?		+CWSAP_CUR=<ssid>,<pwd>,<chl:1-13>,<hidden:0-1>
执行指令 AT+CWSAP_CUR=<ssid>,<pwd>,<chl>,<hidden>	<ssid>：字符串参数 <pwd>：字符串参数 <chl>：整型，1~13 <hidden>： 0：SSID Broadcast 1：Hidden SSID	执行结果
示例： AT+CWSAP_CUR="test_ap","12345678",6,0 正确响应： OK		

4.7. AT+CWLIF

表 4-7. 查看连接上 SoftAP 的客户端

指令	参数	响应
执行指令 AT+CWLIF		+CWLIF: [0] <mac1> +CWLIF: [1] <mac2> ……
示例： AT+CWLIF 正确响应： +CWLIF: [0] e0:2b:e9:8a:46:ac OK		

4.8. AT+CWAUTOCONN

表 4-8. 设置开机是否自动连接 AP

指令	参数	响应
帮助指令 AT+CWAUTOCONN=?		+CWAUTOCONN=<enable>
查询指令 AT+CWAUTOCONN?		+CWAUTOCONN:<enable>
执行指令 AT+CWAUTOCONN=<enable>	<enable>: 0~1 0: disable auto connect 1: enable auto connect	执行结果
示例： AT+CWAUTOCONN=1 正确响应： OK		
补充说明： +CWAUTOCONN 设置为 1 后，连接 AP 成功就会将 AP 信息保存到 FLASH 中，重启后就会自动根据 FLASH 中存储的 AP 信息连接 AP。		

4.9. AT+CIPAP

表 4-9. 查询或设置 SoftAP 的 IP 地址

指令	参数	响应
帮助指令 AT+CIPAP=?		+CIPAP=<"ip">[,<"gateway">,<"netmask">]
查询指令 AT+CIPAP?	<"ip">：SoftAP 的 IPv4 地址，字符串参数 <"gateway">：网关	+CIPAP:ip:<"ip"> +CIPAP:gateway:<"gateway"> +CIPAP:netmask:<"netmask">

指令	参数	响应
	<"netmask">：子网掩码 <"ipv6 addr">：SoftAP 的 IPv6 地址	+CIPAP:ip6ll:<"ipv6 addr">
执行指令 AT+CIPAP=<"ip">[,<"gateway">,<"netmask">]		执行结果
示例： AT+CIPAP="192.168.2.1","192.168.2.1","255.255.255.0" 正确响应： OK 注：本设置命令仅用于设置 IPv4 网络，不用于设置 IPv6 网络		

5. AT TCPIP 指令集

5.1. AT+PING

表 5-1. Ping 功能

指令	参数	响应
帮助指令 AT+PING=?		+PING=<ip or domain name>
执行指令 AT+PING=<ip or domain>	<ip>：字符串，IP 地址或域名	+<delay_time> +<delay_time> ……
示例 1： AT+PING="192.168.0.1" 正确响应 1： +80 +47 +49 +55 +53 OK 示例 2，PING 公网域名时，必须要接入互联网，否则会失败： AT+PING="www.baidu.com" 正确响应 2： +149 +47 +51 +47 +112 OK		

5.2. AT+CIPSTA

表 5-2. 查询或设置本地 STA 的 IP 地址

指令	参数	响应
帮助指令 AT+CIPSTA=?		+CIPSTA=<ip>,<netmask>,<gw>
查询指令 AT+CIPSTA?		+CIPSTA:<ip> +CIPSTA:<netmask> +CIPSTA:<gw>
执行指令 AT+CIPSTA=<ip>,<netmask>,<gw>	<ip>：字符串参数 <netmask>：字符串参数 <gw>：字符串参数	执行结果

指令	参数	响应
示例 1： AT+CIPSTA? 正确响应 1： +CIPSTA: 192.168.185.45 +CIPSTA: 255.255.255.0 +CIPSTA: 192.168.185.1 OK 示例 2： AT+CIPSTA="192.168.185.45","255.255.255.0","192.168.185.1" 正确响应 2： OK		

5.3. AT+CIPSTART

表 5-3. 建立 TCP 连接或 UDP 传输

指令	参数	响应
帮助指令 AT+CIPSTART=?		+CIPSTART=[<con_id>,]<type:TCP or UDP>,<remote ip>,<remote port>[,udp local port][,tcp keep alive]
执行指令 AT+CIPSTART=[<con_id>,]<type>,<remote ip>,<remote port>, [udp local port] , [tcp keep alive]	[<con_id>]：连接 ID，整型。用于多连接模式。 <type>：“TCP” or “UDP”，字符串参数 <remote ip>：Server IP，字符串参数 <remote port>：Server Port，整型 [udp local port]：绑定本机的端口值，整型。用于 UDP 传输。 [tcp keep alive]：keepalive 时间，整型，单位：秒。用于 TCP 连接。	执行结果 <con_id>,OK 或者 ERROR
示例 1，单连接模式下，建立 TCP 连接： AT+CIPSTART="TCP","192.168.0.2",2001,60 正确响应： 0,OK 示例 2，多连接模式下，建立 TCP 连接： AT+CIPSTART=2,"TCP","192.168.0.2",2001,60 正确响应： 2,OK		

示例 3，单连接模式下，建立 UDP 传输： AT+CIPSTART="UDP", "192.168.0.2",5001 正确响应： 0,OK 示例 4，多连接模式下，建立 UDP 传输，指定本地端口号 8888： AT+CIPSTART=3,"UDP", "192.168.0.2",5001,8888 正确响应： 3,OK 注：该项测试需要在测试机上运行 sokit 工具。多连接模式下，必须指定 con_id。

5.4. AT+CIPSEND

表 5-4. 发送数据

指令	参数	响应
帮助指令 AT+CIPSEND=?		Usage: Normal Mode Usage: +CIPSEND=[con_id,]<len>[,<remote ip>,<remote port>] PassThrough Mode Usage: +CIPSEND
进入普通传输模式， 执行指令 AT+CIPSEND=[con_id,]<len>,[<remote ip>, <remote port>]	[con_id]：连接 ID，整型 <len>：<= 2048，发送长度，整型 [remote ip]：对端 IP，字符串参数 [remote port]：对端端口，整型	><input from keyboard> SEND OK
进入 WiFi 透传模式，执行指令 AT+CIPSEND		OK > <input from keyboad>
示例 1，单连接模式下，可省略 con_id： AT+CIPSEND=10 正确响应 1： > SEND OK OK 示例 2，多连接模式下，需要指定 con_id： AT+CIPSEND=1,10 正确响应 2： > SEND OK		

OK

示例 3，UDP 传输可以指定对端 IP 和端口：

AT+CIPSEND=1,20,"192.168.0.2",5001

正确响应 3：

>

SEND OK

OK

示例4：GD32VW553作为TCP客户端，建立单连接，实现UART Wi-Fi透传

连接到路由器

AT+CWJAP="test_ap","1234567890"

查询GD32VW553设备IP地址，以192.168.1.27为例

AT+CIPSTA?

PC测试机与GD32VW553设备连接到同一个路由器，并运行网络调试工具，创建一个TCP服务器。例如IP地址为192.168.1.2，端口号为5678。GD32VW553连接该TCP服务器

AT+CIPSTART="TCP","192.168.1.2",5678,0

进入透传接收模式

AT+CIPMODE=1

进入透传发送模式，并发送数据

AT+CIPSEND

OK

>

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，再发下一条命令。

+++

退出UART WiFi透传接收模式

AT+CIPMODE=0

关闭TCP连接

AT+CIPCLOSE

注：

进入 WiFi 透传模式，GD32VW553 设备每次最大接收 8192 字节，最大发送 2920 字节。如果收到的数据长度大于等于 2920 字节，数据会被分为 2920 字节一组的块进行发送，否则会等待 20 毫秒（您可以通过 AT+TRANSINTVL 命令配置此间隔）或等待收到的数据大于等于 2920 字节再发送数据。当输入单独一包+++时，退出透传模式下的数据发送模式，请至少间隔 1 秒再发送下一条 AT 命令。

AT+CIPSEND 命令必须在开启透传模式以及单连接下使用。若为 WiFi-UDP 透传，AT+CIPSTART 命令的<udp local port>必须指定。

该项测试需要在测试机上运行 sokit 或其他网络测试工具。

透传模式仅支持 TCP 单连接和 UDP 固定通信对端的情况。

5.5. AT+CIPSERVER

表 5-5. 建立 TCP/UDP 服务器

指令	参数	响应
帮助指令 AT+CIPSERVER=?		+CIPSERVER=<mode:0-1>[,<type>,<port>]
执行指令 AT+CIPSERVER=<mode>,[,<type>,<port>]	<mode>： 0：关闭服务器 1：建立服务器 <type>： “TCP”or“UDP”，字符串参数 <port>： 可选参数，整型	执行结果
示例 1：建立服务器时，必须指定<type>,<port>： AT+CIPSERVER=1,"TCP",5004 正确响应： OK 示例2：关闭服务器，同时会将所有连接关闭： AT+CIPSERVER=0 正确响应： OK 注： 1、多连接模式下（AT+CIPMUX=1）才能建立服务器。 2、服务器只允许建立一个。		

5.6. AT+CIPCLOSE

表 5-6. 关闭 TCP 连接或 UDP 传输

指令	参数	响应
帮助指令 AT+CIPCLOSE=?		+CIPCLOSE=[con_id]
执行指令 AT+CIPCLOSE=[con_id]	[con_id]：连接 ID，整型	执行结果
示例 1： AT+CIPCLOSE=2 正确响应 CLOSED 2 OK 示例 2，单连接模式下，可省略 con_id：		

AT+CIPCLOSE 正确响应 CLOSED OK 示例 3，当 con_id 设置为 MAX_CLIENT_NUM（默认为 9）时，关闭所有连接： AT+CIPCLOSE=9 正确响应 CLOSED OK

5.7. AT+CIPSTATUS

表 5-7. 查询 Wi-Fi 连接信息

指令	参数	响应
执行指令 AT+CIPSTATUS		STATUS: 5
示例： AT+CIPSTATUS 正确响应： STATUS: 2 OK		
补充说明：STATUS 2：STA 已和 AP 建立连接并且获得 IP 地址 3：STA 已建立 TCP 连接或 UDP 传输客户端 4：DHCP 处理中 5：其他状态		

5.8. AT+CIFSR

表 5-8. 查询本地 IP 地址信息

指令	参数	响应
执行指令 AT+CIFSR		+CIFSR:APIP,<ip> +CIFSR:APMAC,<mac> Or +CIFSR:STAIP,<ip> +CIFSR:STAMAC,<mac>
示例： AT+CIFSR 正确响应： +CIFSR:STAIP,192.168.2.3 +CIFSR:STAMAC,76:ba:ed:20:22:a2		

指令	参数	响应
OK		

5.9. AT+CIPMODE

表 5-9. 查询或设置传输模式

指令	参数	响应
帮助指令 AT+CIPMODE=?		+CIPMODE=<mode:0-1>
查询指令 AT+CIPMODE?		当前传输模式 +CIPMODE:<mode>
执行指令 AT+CIPMODE=<mode>	<mode>：传输模式 0: 正常传输模式 1: WiFi 透传接收模式	执行结果
示例： AT+CIPMODE=1 正确响应： OK 注： WiFi 透传接收模式，仅支持 TCP 单连接、UDP 固定通信对端的情况。 WiFi 透传接收模式，每次接收的数据最大长度是 2920 字节。		

5.10. AT+CIPMUX

表 5-10. 查询或启用/禁用多连接模式

指令	参数	响应
帮助指令 AT+CIPMUX=?		+CIPMUX=<mode:0-1>
查询指令 AT+CIPMUX?		当前连接模式 +CIPMUX:<mode>
执行指令 AT+CIPMUX=<mode>	<mode>：连接模式 0: 单连接模式 1: 多连接模式	执行结果
示例： AT+CIPMUX=1 正确响应： OK 注： 当所有连接都断开时才可以更改连接模式。		

指令	参数	响应
如果建立了 TCP/UDP 服务器，需要关闭服务器才可以更改连接模式。		

5.11. AT+CIPSTATE

表 5-11. 查询 TCP/UDP 连接信息

指令	参数	响应
执行指令 AT+CIPSTATE		存在连接时返回连接信息： +CIPSTATE: <con_id>,<type>,<remote ip>,<remote port>,<local port>,<fd>,<role> 没有连接时返回： OK
示例： AT+CIPSTATE 正确响应： +CIPSTATE:3,TCP,192.168.19.115,5000,51955,3,0 OK		
补充说明： type：“TCP” or “UDP”。 role：0，模块作为客户端；1，模块作为服务器。		

5.12. AT+CIPDOMAIN

表 5-12. 域名解析

指令	参数	响应
帮助指令 AT+CIPDOMAIN=?		+CIPDOMAIN=<"domain name">[,<ip network>]
执行指令 AT+CIPDOMAIN =<"domain name">[,<ip network>]	<"domain name">：域名，字符串参数 [,<ip network>]：首选 IP 类型，整型，默认值为 1。 1：首选解析为 IPv4 地址 2：只解析为 IPv4 地址 3：只解析为 IPv6 地址	执行结果 +CIPDOMAIN:<IP>
示例： AT+CIPDOMAIN="www.baidu.com",1 正确响应： +CIPDOMAIN:<"36.152.44.132"> OK 注：		

如果需要解析公网域名，需要先接入互联网。

5.13. AT+CIPSNTPCFG

表 5-13. 查询或设置时区和 SNTP 服务器

指令	参数	响应
帮助指令 AT+CIPSNTPCFG=?		+CIPSNTPCFG:<enable>,<timezone>,[<SNTP server1>,<SNTP server2>,<SNTP server3>]
查询指令 AT+CIPSNTPCFG?		+CIPSNTPCFG: <enable>,<timezone>,[<SNTP server1>,<SNTP server2>,<SNTP server3>]
执行指令 AT+CIPSNTPCFG=<enable>,<timezone>,[<SNTP server1>,<SNTP server2>,<SNTP server3>]	<enable>： 0：禁用 SNTP 服务 1：使能 SNTP 服务 <timezone>：时区 格式：hour|minute, hour:-12~+14, minute:00~59 [<SNTP server1>]： SNTP 服务器，字符串参数	执行结果
示例： AT+CIPSNTPCFG=1,800,"cn.pool.ntp.org" 正确响应： OK 示例： AT+CIPSNTPCFG=1,-1200 正确响应： OK 示例： AT+CIPSNTPCFG=0,800 正确响应： OK 注： 使能 SNTP 服务必须接入互联网。 使能 SNTP 服务后，时间同步后会打印+TIME_UPDATED。		

5.14. AT+CIPSNTPTIME

表 5-14. 查询 SNTP 时间

指令	参数	响应
执行指令 AT+CIPSNTPTIME		已同步时间： SNTP time: <SNTP time> 未同步时间： Please start the SNTP or wait for the SNTP time update
示例： AT+CIPSNTPTIME 正确响应： SNTP time: 2025-09-02 Tuesday 15:38:14 OK		
补充说明： 需要使能 SNTP 服务。 等待+TIME_UPDATED 打印后才能获取同步时间。		

5.15. AT+CIPSNTPINTV

表 5-15. 查询或设置 SNTP 时间同步间隔

指令	参数	响应
帮助指令 AT+CIPSNTPINTV=?		+CIPSNTPINTV=<interval second>
查询指令 AT+CIPSNTPINTV?		+CIPSNTPINTV:<interval second>
执行指令 AT+CIPSNTPINTV=<interval second>	<interval second>： SNTP 时间同步间隔，整型，单位：秒，范围：[15，4294967]	执行结果
示例： AT+CIPSNTPINTV? 正确响应： +CIPSNTPINTV:20 OK 示例： AT+CIPSNTPINTV=20 正确响应： OK		

5.16. AT+WSCFG

表 5-16. 设置 WebSocket 参数

指令	参数	响应
帮助指令 AT+WSCFG=?		+WSCFG=<link_id>,<ping_intv_sec>,<ping_timeout_sec>[,<buffer_size>]
执行指令 AT+WSCFG=<link_id>,<ping_intv_sec>,<ping_timeout_sec>[,<buffer_size>]	<link_id>：WebSocket 连接 ID，整型，范围：0~2，即最大支持三个 WebSocket 连接 <ping_intv_sec>：发送 WebSocket Ping 间隔，整型，单位：秒，范围：1~7200，默认值：10，即每隔 10 秒发送一次 WebSocket Ping 包 <ping_timeout_sec>：WebSocket Ping 超时，整型，单位：秒，范围：1~7200，默认值：120，即 120 秒未收到 WebSocket Pong 包，则关闭连接 <buffer_size>：WebSocket 缓冲区大小，整型，单位：字节，范围：1~8192，默认值：1024	执行结果
示例： AT+WSCFG=0,15,120,1024 正确响应： OK 注： 请在 AT+WSOPEN 命令之前设置本命令，否则本命令不会生效。		

5.17. AT+WSHEAD

表 5-17. 查询或设置 WebSocket 请求头

指令	参数	响应
帮助指令 AT+WSHEAD=?		+WSHEAD=<req_header_len>
查询指令 AT+WSHEAD?	<index>：WebSocket 请求头的索引值，整型 <"req_header">：WebSocket 请求头，字符串参数	+WSHEAD:<index>,<"req_header"> ...

指令	参数	响应
执行指令 AT+WSHEAD=<req_header_len>	<req_header_len>：WebSocket 请求头长度，整型，单位：字节 0：清除所有已设置的 WebSocket 请求头 其他值：设置一个新的 WebSocket 请求头	执行结果 OK > <input from keyboad>
示例 1： AT+WSHEAD=31 正确响应 1： OK > OK 示例 2： AT+WSHEAD? 正确响应 2： +WSHEAD:0,"Sec-WebSocket-Protocol: chat-v2" +WSHEAD:1,"Cookie: session_id=abc123" OK 注： 本命令一次只能设置一个 WebSocket 请求头，但可多次调用本命令，支持最多 5 个不同的 WebSocket 请求头。 WebSocket 请求头的形式为：key: value。 本命令设置的 WebSocket 请求头是全局性的，一旦设置，所有 WebSocket 的命令都会携带这些请求头。		

5.18. AT+WSOPEN

表 5-18. 查询或打开一个 WebSocket 连接

指令	参数	响应
帮助指令 AT+WSOPEN=?		+WSOPEN=<link_id>,<"uri">[,<"subprotocol">][,<timeout_ms>][,<"auth">]
查询指令 AT+WSOPEN?	<state>：WebSocket 连接的状态，整型 0：WebSocket 连接已关闭 1：WebSocket 连接正在重连 2：已建立 WebSocket 连接 3：接收 WebSocket Pong 超时或读取连接数据错误，正在等待重连	+WSOPEN:<link_id>,<state>,<"uri"> ...

指令	参数	响应
	4：已收到服务器端 WebSocket 关闭帧，正在发送关闭帧到服务器	
执行指令 AT+WSOPEN=<link_id>,<"uri">[,<"subprotocol">][,<timeout_ms>][,<"auth">]	<link_id>：WebSocket 连接 ID，整型，范围：0~2，即最大支持三个 WebSocket 连接 <"uri">：WebSocket 服务器的统一资源标识符，字符串参数 <"subprotocol">：WebSocket 子协议，字符串参数 <timeout_ms>：建立 WebSocket 连接的超时时间，整型，单位：毫秒，范围：0~180000，默认值：15000 <"auth">：WebSocket 鉴权，字符串参数	执行结果 +WS_CONNECTED:<link_id>
示例 1： AT+WSOPEN=0,"wss://free.blr2.piesocket.com/v3/1?api_key=GzyBsP4EjvVESsQYIDskqlR1MAyZJo8mDgOABlha&notify_self=1" 正确响应 1： +WS_CONNECTED:0 OK 示例 2： AT+WSOPEN? 正确响应 2： +WSOPEN:0,2,"wss://free.blr2.piesocket.com/v3/1?api_key=GzyBsP4EjvVESsQYIDskqlR1MAyZJo8mDgOABlha&notify_self=1" OK 注： 有关参数<"subprotocol">的更多信息请参考 RFC6455 1.9 章节。 有关参数<"auth">的更多信息请参考 RFC6455 4.1.12 章节。		

5.19. AT+WSSEND

表 5-19. 向 WebSocket 连接发送数据

指令	参数	响应
帮助指令 AT+WSSEND=?		+WSSEND=<link_id>,<length>[,<opcode>][,<timeout_ms>]
执行指令	<link_id>：WebSocket 连接	执行结果

指令	参数	响应
AT+WSSEND=<link_id>,<length>[,<opcode>][,<timeout_ms>]	ID，整型，范围：0~2，即最大支持三个 WebSocket 连接 <length>：发送的数据长度，整型，单位：字节，可发送的最大长度为 AT+WSCFG 中的<buffer_size>值减去 10 和系统可分配的堆空间大小的小值 <opcode>：发送的 WebSocket 帧中的 opcode，整型，范围：0~0xF，默认值：1，即 text 帧 0x0：continuation 帧 0x1：text 帧 0x2：binary 帧 0x3-0x7：为其它非控制帧保留 0x8：连接关闭帧 0x9：ping 帧 0xA：pong 帧 0xB-0xF：为其它控制帧保留 <timeout_ms>：发送超时时间，整型，单位：毫秒，范围：0~60000，默认值：10000	OK > <input from keyboad> 若数据传输成功，AT 返回： SEND OK 若未建立连接或数据传输时连接被断开，AT 返回： ERROR
示例： AT+WSSEND=0,5 正确响应： OK > SEND OK 注： 有关参数<opcode>的更多信息请参考 RFC6455 5.2 章节。		

5.20. AT+WSCLOSE

表 5-20. 关闭 WebSocket 连接

指令	参数	响应
帮助指令 AT+WSCLOSE=?		+WSCLOSE=<link_id>
执行指令 AT+WSCLOSE=<link_id>	<link_id>：WebSocket 连接 ID，整型，范围：0~2，即最大支持三个 WebSocket 连接	执行结果
示例：		

指令	参数	响应
AT+WSCLOSE=0 正确响应： OK		

6. AT MQTT 指令集

6.1. AT+MQTTUSERCFG

表 6-1. 设置 MQTT 用户属性

指令	参数	响应
帮助指令 AT+MQTTUSERCFG=?		+MQTTUSERCFG=<LinkID>,<scheme>,<"client_id">,<"username">,<"password">,<cert_key_ID>,<CA_ID>
执行指令 AT+MQTTUSERCFG=<LinkID>,<scheme>,<"client_id">,<"username">,<"password">,<cert_key_ID>,<CA_ID>	<LinkID>：仅支持 link ID 0，整型 <scheme>：连接方案 1：MQTT over TCP 2：MQTT over TLS（不校验证书） 3：MQTT over TLS（校验 server 证书） 4：MQTT over TLS（提供 client 证书） 5：MQTT over TLS（校验 server 证书并且提供 client 证书） <"client_id">：客户端 ID，字符串参数，最大长度 256 字节 <"username">：登录 MQTT broker 的用户名，字符串参数，最大长度 64 字节 <"password">：登录 MQTT broker 的密码，字符串参数，最大长度 64 字节 <cert_key_ID>：证书 ID，仅支持一套 cert 证书，参数为 0，整型 <CA_ID>：CA ID，仅支持一套 CA 证书，参数为 0，整型	执行结果
示例 1： AT+MQTTUSERCFG=0,3,"Gigadevice","user","123456",0,0 正确响应 1： OK		

6.2. AT+MQTTLONGCLIENTID

表 6-2. 设置 MQTT 客户端 ID

指令	参数	响应
帮助指令 AT+MQTTLONGCLIENTID=?		+MQTTLONGCLIENTID=<LinkID>,<length>
执行指令 AT+MQTTLONGCLIENTID=<LinkID>,<length>	<LinkID>：仅支持 link ID 0，整型 <length>：MQTT 客户端 ID 长度。整型，范围：1~1024	执行结果 OK > <input from keyboad>
示例 1： AT+MQTTLONGCLIENTID=0,10 正确响应 1： OK > OK 注： AT+MQTTUSERCFG 命令和 AT+MQTTLONGCLIENTID 命令均可以设置 MQTT 客户端 ID，两者之间的差别为： AT+MQTTUSERCFG 命令受 AT 命令总长度的限制，可设置的客户端 ID 相对较短，而 AT+MQTTLONGCLIENTID 命令可以用来设置相对较长的客户端 ID； 需先设置 AT+MQTTUSERCFG 再使用 AT+MQTTLONGCLIENTID。		

6.3. AT+MQTTLONGUSERNAME

表 6-3. 设置 MQTT 登录用户名

指令	参数	响应
帮助指令 AT+MQTTLONGUSERNAME=?		+MQTTLONGUSERNAME=<LinkID>,<length>
执行指令 AT+MQTTLONGUSERNAME=<LinkID>,<length>	<LinkID>：仅支持 link ID 0，整型 <length>：MQTT 用户名长度。整型，范围：1~1024	执行结果 OK > <input from keyboad>
示例 1： AT+MQTTLONGUSERNAME=0,8 正确响应 1： OK >		

OK 注： AT+MQTTUSERCFG 命令和 AT+MQTTLONGUSERNAME 命令均可以设置 MQTT 用户名，两者之间的差别为： AT+MQTTUSERCFG 命令受 AT 命令总长度的限制，可设置的用户名相对较短，而 AT+MQTTLONGUSERNAME 命令可以用来设置相对较长的用户名； 需先设置 AT+MQTTUSERCFG 再使用 AT+MQTTLONGUSERNAME。

6.4. AT+MQTTLONGPASSWORD

表 6-4. 设置 MQTT 登录密码

指令	参数	响应
帮助指令 AT+MQTTLONGPASSWORD=?		+MQTTLONGPASSWORD=<LinkID>,<length>
执行指令 AT+MQTTLONGPASSWORD=<LinkID>,<length>	<LinkID>：仅支持 link ID 0，整型 <length>：MQTT 密码长度。整型，范围：1~1024	执行结果 OK > <input from keyboad>
示例 1： AT+MQTTLONGPASSWORD=0,12 正确响应 1： OK > OK 注： AT+MQTTUSERCFG 命令和 AT+MQTTLONGPASSWORD 命令均可以设置 MQTT 密码，两者之间的差别为： AT+MQTTUSERCFG 命令受 AT 命令总长度的限制，可设置的密码相对较短，而 AT+MQTTLONGPASSWORD 命令可以用来设置相对较长的密码； 需先设置 AT+MQTTUSERCFG 再使用 AT+MQTTLONGPASSWORD。		

6.5. AT+MQTTCONNCFG

表 6-5. 设置 MQTT 连接属性

指令	参数	响应
帮助指令 AT+MQTTCONNCFG=?		+MQTTCONNCFG=<LinkID>,<keepalive>,<disable_clean_session>,<"lwt_topic">,<"lwt_msg">,<lwt_qos>,<lwt_retain>

执行指令 AT+MQTTCONNCFG=<LinkID>,<keepalive>,<disable_clean_session>,<"lwt_topic">,<"lwt_msg">,<lwt_qos>,<lwt_retain>	<LinkID>：仅支持 link ID 0，整型 <keepalive>：MQTT ping 超时时间，整型，单位：秒，范围：0~7200，默认值：0，会被强制改为 120 秒 <disable_clean_session>：设置 MQTT 清理会话标志 0：使能清理会话 1：禁用清理会话 <"lwt_topic">：遗嘱 topic，字符串参数，最大长度 128 字节 <"lwt_msg">：遗嘱 message，字符串参数，最大长度 128 字节 <lwt_qos>：遗嘱 QoS，整型，范围：0~2，默认值：0 <lwt_retain>：遗嘱 retain，整型，范围：0~1，默认值：0	执行结果
示例 1： AT+MQTTCONNCFG=0,0,0,"will_topic","will_message",0,0 正确响应 1： OK 注： 有关参数<disable_clean_session>的更多信息请参考 MQTT 3.1.1 协议中的 Clean Session 章节。		

6.6. AT+MQTTCONN

表 6-6. 连接 MQTT Broker 或查询 MQTT 状态

指令	参数	响应
帮助指令 AT+MQTTCONN=?		+MQTTCONN=<LinkID>,<"host">,<port>,<reconnect>
查询指令 AT+MQTTCONN?	<LinkID>：仅支持 link ID 0，整型 <state>：MQTT 状态 0：MQTT 未初始化 1：已设置 AT+MQTTUSERCFG 2：已设置 AT+MQTTCONNCFG 3：连接已断开 4：已建立连接 5：已连接，但未订阅 topic	+MQTTCONN:<LinkID>,<state>,<scheme>,<"host">,<port>,<reconnect>

指令	参数	响应
	6：已连接，已订阅过 topic <scheme>：连接方案 1：MQTT over TCP 2：MQTT over TLS（不校验证书） 3：MQTT over TLS（校验 server 证书） 4：MQTT over TLS（提供 client 证书） 5：MQTT over TLS（校验 server 证书并且提供 client 证书） <"host">：MQTT broker 域名，字符串参数，最大长度 128 字节 <port>：MQTT broker 端口，整型，最大端口 65535 <reconnect>：重连 0：MQTT 不自动重连 1：MQTT 自动重连，会消耗较多的内存资源	
执行指令 AT+MQTTCONN=<LinkID>,<"host">,<port>,<reconnect>		执行结果 +MQTTCONNECTED:<LinkID>,<scheme>,<"host">,<port>,<reconnect>
示例 1： AT+MQTTCONN? 正确响应 1： +MQTTCONN:0,0 OK 示例 2： AT+MQTTCONN? 正确响应 2： +MQTTCONN:0,1,3 OK 示例 3： AT+MQTTCONN? 正确响应 3： +MQTTCONN:0,5,3,"192.168.43.50",8883,0 OK		

指令	参数	响应
示例 4： AT+MQTTCONN=0,"192.168.43.50",8883,0 正确响应 4： +MQTTCONNECTED:0,3,"192.168.43.50",8883,0 OK 注： 当 MQTT 连接建立时，会提示 +MQTTCONNECTED:<LinkID>,<scheme>,<"host">,<port>,<reconnect> 消息。 如果 MQTT 不自动重连并且建立连接后又断开，或者自动重连但建立连接后断开并且重连次数达到上限后，则无法再次使用本命令重新建立连接，需要先发送 AT+MQTTCLEAN=0 命令清理信息，重新配置参数，再建立新的连接。 使用 Mosquitto 作为 MQTT broker。		

6.7. AT+MQTTPUB

表 6-7. 发布 MQTT 消息（字符串）

指令	参数	响应
帮助指令 AT+MQTTPUB=?		+MQTTPUB=<LinkID>,<"topic">,<"data">,<qos>,<retain>
执行指令 AT+MQTTPUB=<LinkID>,<"topic">,<"data">,<qos>,<retain>	<LinkID>：仅支持 link ID 0，整型 <"topic">：MQTT topic，字符串参数，最大长度 128 字节 <"data">：MQTT 字符串消息，字符串参数 <qos>：发布消息的 QoS，整型，范围：0~2，默认值：0 <retain>：发布 retain，整型，范围：0~1，默认值：0	执行结果
示例 1： AT+MQTTPUB=0,"topic_test","helloworld",0,0 正确响应 1： OK 注： 本命令不能发送数据\0，若需要发送该数据，请使用 AT+MQTTPUBRAW 命令。		

6.8. AT+MQTTPUBRAW

表 6-8. 发布长 MQTT 消息

指令	参数	响应
帮助指令 AT+MQTTPUBRAW=?		+MQTTPUBRAW=<LinkID>,<"topic">,<length>,<qos>,<retain>
执行指令 AT+MQTTPUBRAW=<LinkID>,<"topic">,<length>,<qos>,<retain>	<LinkID>：仅支持 link ID 0，整型 <"topic">：MQTT topic，字符串参数，最大长度 128 字节 <length>：MQTT 消息长度，整型 <qos>：发布消息的 QoS，整型，范围：0~2，默认值：0 <retain>：发布 retain，整型，范围：0~1，默认值：0	执行结果 OK > <input from keyboad> 若串口数据传输成功，AT 返回： +MQTTPUB:OK 若串口数据传输失败，AT 返回： +MQTTPUB:FAIL
示例 1： AT+MQTTPUBRAW=0,"topic_test",9,0,0 正确响应 1： OK > +MQTTPUB:OK OK		

6.9. AT+MQTTSUB

表 6-9. 订阅 MQTT Topic 或查询已订阅 Topic

指令	参数	响应
帮助指令 AT+MQTTSUB=?		+MQTTSUB=<LinkID>,<"topic">,<qos>
查询指令 AT+MQTTSUB?	<LinkID>：仅支持 link ID 0，整型 <state>：MQTT 状态 0：MQTT 未初始化 1：已设置 AT+MQTTUSERCFG 2：已设置 AT+MQTTCONNCFG 3：连接已断开 4：已建立连接 5：已连接，但未订阅 topic 6：已连接，已订阅过 topic <"topic">：订阅的 topic，字符串	+MQTTSUB:<LinkID>,<state>,<"topic1">,<qos> +MQTTSUB:<LinkID>,<state>,<"topic2">,<qos> +MQTTSUB:<LinkID>,<state>,<"topic3">,<qos> ...

指令	参数	响应
	参数 <qos>：订阅的 QoS，整型，范围：0~2	
执行指令 AT+MQTTSUB=<LinkID>,<"topic">,<qos>		执行结果 若未订阅过该 topic 且订阅成功，AT 返回： OK 若已订阅过该 topic，AT 返回： ALREADY SUBSCRIBE
示例 1： AT+MQTTSUB? 正确响应 1： +MQTTSUB:0,6,"topic_test1",0 +MQTTSUB:0,6,"topic_test",0 OK 示例 2： AT+MQTTSUB=0,"topic_test",0 正确响应 2： OK 示例 3： AT+MQTTSUB=0,"topic_test",0 正确响应 3： ALREADY SUBSCRIBE OK 注： 可多次调用本命令，以订阅多个不同的 topic。 当 AT 接收到已订阅的 topic 的 MQTT 消息时，返回： +MQTTSUBRECV:<LinkID>,<"topic">,<data_length>,data。		

6.10. AT+MQTTUNSUB

表 6-10. 取消订阅 MQTT Topic

指令	参数	响应
帮助指令 AT+MQTTUNSUB=?		+MQTTUNSUB=<LinkID>,<"topic">
执行指令 AT+MQTTUNSUB=<LinkID>,<"topic">	<LinkID>：仅支持 link ID 0，整型 <"topic">：MQTT topic，字符串	执行结果 若已订阅过该 topic 且取消订阅成功，AT 返回：

	参数，最大长度 128 字节	OK 若未订阅过该 topic，AT 返回： NO UNSUBSCRIBE
示例 1： AT+MQTTUNSUB=0,"topic_test" 正确响应 1： OK 示例 2： AT+MQTTUNSUB=0,"topic_test" 正确响应 2： NO UNSUBSCRIBE OK 注： 可多次调用本命令，以取消订阅不同的 topic。		

6.11. AT+MQTTCLEAN

表 6-11. 断开 MQTT 连接并释放资源

指令	参数	响应
帮助指令 AT+MQTTCLEAN=?		+MQTTCLEAN=<LinkID>
执行指令 AT+MQTTCLEAN=<LinkID>	<LinkID>：仅支持 link ID 0，整型	执行结果
示例 1： AT+MQTTCLEAN=0 正确响应 1： OK 注： 当 MQTT client 被动断开连接，如网络断开或 MQTT broker 关闭等原因导致 MQTT 连接意外断开时，会提示+MQTTDISCONNECTED:<LinkID>消息。		

6.12. AT MQTT 错误码

表 6-12. AT MQTT 错误码

错误类型	错误码
AT_MQTT_NO_CONFIGURED	0x6001
AT_MQTT_NOT_IN_CONFIGURED_STATE	0x6002
AT_MQTT_UNINITIATED_OR_ALREADY_CLEAN	0x6003

AT_MQTT_ALREADY_CONNECTED	0x6004
AT_MQTT_MALLOC_FAILED	0x6005
AT_MQTT_NULL_LINK	0x6006
AT_MQTT_NULL_PARAMTER	0x6007
AT_MQTT_PARAMETER_COUNTS_IS_WRONG	0x6008
AT_MQTT_TLS_CONFIG_ERROR	0x6009
AT_MQTT_PARAM_PREPARE_ERROR	0x600A
AT_MQTT_CLIENT_START_FAILED	0x600B
AT_MQTT_CLIENT_PUBLISH_FAILED	0x600C
AT_MQTT_CLIENT_SUBSCRIBE_FAILED	0x600D
AT_MQTT_CLIENT_UNSUBSCRIBE_FAILED	0x600E
AT_MQTT_CLIENT_DISCONNECT_FAILED	0x600F
AT_MQTT_LINK_ID_READ_FAILED	0x6010
AT_MQTT_LINK_ID_VALUE_IS_WRONG	0x6011
AT_MQTT_SCHEME_READ_FAILED	0x6012
AT_MQTT_SCHEME_VALUE_IS_WRONG	0x6013
AT_MQTT_CLIENT_ID_READ_FAILED	0x6014
AT_MQTT_CLIENT_ID_IS_NULL	0x6015
AT_MQTT_CLIENT_ID_IS_OVERLENGTH	0x6016
AT_MQTT_USERNAME_READ_FAILED	0x6017
AT_MQTT_USERNAME_IS_NULL	0x6018
AT_MQTT_USERNAME_IS_OVERLENGTH	0x6019
AT_MQTT_PASSWORD_READ_FAILED	0x601A
AT_MQTT_PASSWORD_IS_NULL	0x601B
AT_MQTT_PASSWORD_IS_OVERLENGTH	0x601C
AT_MQTT_CERT_KEY_ID_READ_FAILED	0x601D
AT_MQTT_CERT_KEY_ID_VALUE_IS_WRONG	0x601E
AT_MQTT_CA_ID_READ_FAILED	0x601F
AT_MQTT_CA_ID_VALUE_IS_WRONG	0x6020
AT_MQTT_CA_LENGTH_ERROR	0x6021
AT_MQTT_CA_READ_FAILED	0x6022
AT_MQTT_CERT_LENGTH_ERROR	0x6023
AT_MQTT_CERT_READ_FAILED	0x6024
AT_MQTT_KEY_LENGTH_ERROR	0x6025
AT_MQTT_KEY_READ_FAILED	0x6026
AT_MQTT_PATH_READ_FAILED	0x6027
AT_MQTT_PATH_IS_NULL	0x6028
AT_MQTT_PATH_IS_OVERLENGTH	0x6029
AT_MQTT_VERSION_READ_FAILED	0x602A
AT_MQTT_KEEPALIVE_READ_FAILED	0x602B
AT_MQTT_KEEPALIVE_IS_NULL	0x602C
AT_MQTT_KEEPALIVE_VALUE_IS_WRONG	0x602D

AT_MQTT_DISABLE_CLEAN_SESSION_READ_FAILED	0x602E
AT_MQTT_DISABLE_CLEAN_SESSION_VALUE_IS_WRONG	0x602F
AT_MQTT_LWT_TOPIC_READ_FAILED	0x6030
AT_MQTT_LWT_TOPIC_IS_NULL	0x6031
AT_MQTT_LWT_TOPIC_IS_OVERLENGTH	0x6032
AT_MQTT_LWT_MSG_READ_FAILED	0x6033
AT_MQTT_LWT_MSG_IS_NULL	0x6034
AT_MQTT_LWT_MSG_IS_OVERLENGTH	0x6035
AT_MQTT_LWT_QOS_READ_FAILED	0x6036
AT_MQTT_LWT_QOS_VALUE_IS_WRONG	0x6037
AT_MQTT_LWT_RETAIN_READ_FAILED	0x6038
AT_MQTT_LWT_RETAIN_VALUE_IS_WRONG	0x6039
AT_MQTT_HOST_READ_FAILED	0x603A
AT_MQTT_HOST_IS_NULL	0x603B
AT_MQTT_HOST_IS_OVERLENGTH	0x603C
AT_MQTT_PORT_READ_FAILED	0x603D
AT_MQTT_PORT_VALUE_IS_WRONG	0x603E
AT_MQTT_RECONNECT_READ_FAILED	0x603F
AT_MQTT_RECONNECT_VALUE_IS_WRONG	0x6040
AT_MQTT_TOPIC_READ_FAILED	0x6041
AT_MQTT_TOPIC_IS_NULL	0x6042
AT_MQTT_TOPIC_IS_OVERLENGTH	0x6043
AT_MQTT_DATA_READ_FAILED	0x6044
AT_MQTT_DATA_IS_NULL	0x6045
AT_MQTT_DATA_IS_OVERLENGTH	0x6046
AT_MQTT_QOS_READ_FAILED	0x6047
AT_MQTT_QOS_VALUE_IS_WRONG	0x6048
AT_MQTT_RETAIN_READ_FAILED	0x6049
AT_MQTT_RETAIN_VALUE_IS_WRONG	0x604A
AT_MQTT_PUBLISH_LENGTH_READ_FAILED	0x604B
AT_MQTT_PUBLISH_LENGTH_VALUE_IS_WRONG	0x604C
AT_MQTT_RECV_LENGTH_IS_WRONG	0x604D
AT_MQTT_CREATE_SEMA_FAILED	0x604E
AT_MQTT_CREATE_EVENT_GROUP_FAILED	0x604F
AT_MQTT_URI_PARSE_FAILED	0x6050
AT_MQTT_IN_DISCONNECTED_STATE	0x6051
AT_MQTT_HOSTNAME_VERIFY_FAILED	0x6052
注： MQTT 错误码以 ERR CODE:0x<%08x>形式打印。	

7. AT HTTP 指令集

7.1. AT+HTTPCLIENT

表 7-1. 发送 HTTP 客户端请求

指令	参数	响应
帮助指令 AT+HTTPCLIENT=?		+HTTPCLIENT=<method>,<content-type>,<"url">,[<"host">],[<"path">],<transport_type>[,<"data">][,<"http_req_header">][,<"http_req_header">][...]
执行指令 AT+HTTPCLIENT=<method>,<content-type>,<"url">,[<"host">],[<"path">],<transport_type>[,<"data">][,<"http_req_header">][,<"http_req_header">][...]	<method>：HTTP 客户端请求方法，整型 1：HEAD 2：GET 3：POST <content-type>：客户端请求数据类型，整形 0：application/x-www-form-urlencoded 1：application/json 2：multipart/form-data 3：text/xml <"url">：HTTP URL，字符串参数，当后面的<host> 和<path>参数为空时，本参数会自动覆盖这两个参数 [<"host">]：域名或 IP 地址，字符串参数 [<"path">]：HTTP 路径，字符串参数 < transport_type>：HTTP 客户端传输类型，整型 1：HTTP OVER TCP 2：HTTP OVER SSL <"data">：当<method> 是 POST 请求时，本参数为发送给 HTTP 服务器的数据。否则，这个参数不存在。整型 [<"http_req_header">]：发送给 server 的请求头，最多可以带 5 个请求头，字符串参数	执行结果

示例 1：HEAD 请求
AT+HTTPCLIENT=1,3,"https://www.baidu.com/","","",2
正确响应 1：
+HTTPCLIENT:277
OK

示例 2：GET 请求
AT+HTTPCLIENT=2,3,"https://www.baidu.com/","","",2
正确响应 2：
+HTTPCLIENT:29506,<!DOCTYPE html>
…(HTTP 资源的具体内容)
OK

示例 3：POST 请求
AT+HTTPCLIENT=3,3,"https://192.168.237.1/portal.html","","",2,"ssid=test&password=12345678"
正确响应 3：
SEND OK

注：
如果包含 URL 的整条命令的长度超过了 256 字节，请先使用 AT+HTTPURLCFG 命令预配置 URL，然后本命令里的<"url">参数需要设置为""。
如果 url 参数不为空，HTTP 客户端将使用它并忽略 host 参数和 path 参数；如果 url 参数被省略或字符串为空，HTTP 客户端将使用 host 参数和 path 参数。
如果包含<"data"> 参数的整条命令的长度超过了 256 字节，请使用 AT+HTTPCPOST 命令。
要设置更多的 HTTP 请求头，请使用 AT+HTTPCHEAD 命令。

7.2. AT+HTTPGETSIZE

表 7-2. 获取 HTTP 资源大小

指令	参数	响应
帮助指令 AT+HTTPGETSIZE=?		+HTTPGETSIZE=<"url">[,<tx size>][,<rx size>][,<timeout>]
执行指令 AT+HTTPGETSIZE=<"url">[,<tx size>][,<rx size>][,<timeout>]	<"url">：HTTP URL，字符串参数 [<tx size>]：HTTP 发送缓存大小，整型，单位：字节，范围：0~10240，默认值：2048 [<rx size>]：HTTP 接收缓存大小，整型，单位：字节，范围：0~10240，默认值：2048 [<timeout>]：网络超时时间，整型，单位：毫秒，范围：	执行结果 +HTTPGETSIZE:<size>

	0~180000，默认值：5000 <size>：HTTP 资源大小，整型	
示例 1： AT+HTTPGETSIZE="https://www.baidu.com",,2048 正确响应 1： +HTTPGETSIZE:277 OK 注： 如果包含 URL 的整条命令的长度超过了 256 字节，请先使用 AT+HTTPURLCFG 命令预配置 URL，然后本命令里的<"url">参数需要设置为""。 若需设置 HTTP 请求头，请使用 AT+HTTPCHEAD 命令设置。		

7.3. AT+HTTPCGET

表 7-3. 获取 HTTP 资源

指令	参数	响应
帮助指令 AT+HTTPCGET=?		+HTTPCGET=<"url">[,<tx size>][,<rx size>][,<timeout>]
执行指令 AT+HTTPCGET=<"url">[,<tx size>][,<rx size>][,<timeout>]	<"url">：HTTP URL，字符串参数 [<tx size>]：HTTP 发送缓存大小，整型，单位：字节，范围：0~10240，默认值：2048 [<rx size>]：HTTP 接收缓存大小，整型，单位：字节，范围：0~10240，默认值：2048 [<timeout>]：网络超时时间，整型，单位：毫秒，范围：0~180000，默认值：5000	执行结果 +HTTPCGET:<size>,<data>
示例 1： AT+HTTPCGET="https://www.baidu.com",4096,4096 正确响应 1： +HTTPCGET:29506,<!DOCTYPE html> …(HTTP 资源的具体内容) OK 注： 如果包含 URL 的整条命令的长度超过了 256 字节，请先使用 AT+HTTPURLCFG 命令预配置 URL，然后本命令里的<"url">参数需要设置为""。 若需设置 HTTP 请求头，请使用 AT+HTTPCHEAD 命令设置。		

7.4. AT+HTTPCPOST

表 7-4. Post 指定长度的 HTTP 数据

指令	参数	响应
帮助指令 AT+HTTPCPOST=?		+HTTPCPOST=<"url">,<length>[,<http_req_header_cnt>][,<http_req_header>..<http_req_header>]
执行指令 AT+HTTPCPOST=<"url">,<length>[,<http_req_header_cnt>][,<http_req_header>..<http_req_header>]	<"url">：HTTP URL，字符串参数 <length>：POST 的 HTTP 数据长度，整型 [<http_req_header_cnt>]：[<http_req_header>] 参数的数量，整型，最大为 5 [<http_req_header>]：发送给 server 的请求头，最多可以带 5 个请求头，字符串参数	执行结果 OK > <input from keyboad> 若 POST 成功，AT 返回： SEND OK 若 POST 失败，AT 返回： SEND FAIL
示例 1： AT+HTTPCPOST="https://192.168.237.1/portal.html",31 正确响应 1： OK > SEND OK 注： 如果包含 URL 的整条命令的长度超过了 256 字节，请先使用 AT+HTTPURLCFG 命令预配置 URL，然后本命令里的<"url"> 参数需要设置为""。 该命令的客户端请求数据类型默认为 application/x-www-form-urlencoded。 若需设置 HTTP 请求头，请使用 AT+HTTPCHEAD 命令设置。		

7.5. AT+HTTPCPUT

表 7-5. Put 指定长度的 HTTP 数据

指令	参数	响应
帮助指令 AT+HTTPCPUT=?		+HTTPCPUT=<"url">,<length>[,<http_req_header_cnt>][,<http_req_header>..<http_req_header>]
执行指令 AT+HTTPCPUT=<"url">,<length>[,<http_req_header_cnt>][,<http_req_header>..<http_req_header>]	<"url">：HTTP URL，字符串参数 <length>：Put 的 HTTP 数据长度，整型 [<http_req_header_cnt>]：	执行结果 OK > <input from keyboad> 若 Put 成功，AT 返回： SEND OK

	[<http_req_header>] 参数的数量，整型，最大为 5 [<http_req_header>]：发送给 server 的请求头，最多可以带 5 个请求头，字符串参数	若 Put 失败，AT 返回： SEND FAIL
示例 1： AT+HTTPCPUT="https://192.168.237.1/portal.html",31 正确响应 1： OK > SEND OK 注： 如果包含 URL 的整条命令的长度超过了 256 字节，请先使用 AT+HTTPURLCFG 命令预配置 URL，然后本命令里的<"url"> 参数需要设置为""。 若需设置 HTTP 请求头，请使用 AT+HTTPCHEAD 命令设置。		

7.6. AT+HTTPURLCFG

表 7-6. 查询或设置长的 HTTP URL

指令	参数	响应
帮助指令 AT+HTTPURLCFG=?		+HTTPURLCFG=<url length>
查询指令 AT+HTTPURLCFG?		[+HTTPURLCFG:<url length>,<data>]
执行指令 AT+HTTPURLCFG=<url length>	<url length>：HTTP URL 长度，整型，单位：字节 0：清除 HTTP URL 配置 8~8192：设置的 HTTP URL 的长度 <data>：HTTP URL 数据，字符串类型	执行结果 OK > <input from keyboad> SET OK
示例 1： AT+HTTPURLCFG=21 正确响应 1： OK > SET OK 示例 2： AT+HTTPURLCFG? 正确响应 2：		

指令	参数	响应
+HTTPURLCFG:21,https://www.baidu.com OK		

7.7. AT+HTTPCHEAD

表 7-7. 查询或设置 HTTP 请求头

指令	参数	响应
帮助指令 AT+HTTPCHEAD=?		+HTTPCHEAD=<req_header_len>
查询指令 AT+HTTPCHEAD?		+HTTPCHEAD:<index>,<"req_header"> >
执行指令 AT+HTTPCHEAD=<req_header_len>	<index>：HTTP 请求头的索引值，整型 <"req_header">：HTTP 请求头，字符串类型 <req_header_len>：HTTP 请求头长度，整型，单位：字节 0：清除所有已设置的 HTTP 请求头 其他值：设置的新的 HTTP 请求头的长度	执行结果 OK > <input from keyboad>
示例 1： AT+HTTPCHEAD=18 正确响应 1： OK > OK 示例 2： AT+HTTPCHEAD? 正确响应 2： +HTTPCHEAD:0,"Range: bytes=0-255" OK 注： HTTP 请求头的形式为 key: value。 本命令一次只能设置一个 HTTP 请求头，但可以多次调用本命令以设置多个不同的 HTTP 请求头。 本命令设置的 HTTP 请求头是全局性的，一旦设置，所有 HTTP 的命令都会携带这些请求头。 本命令设置的 HTTP 请求头中的 key 如果和其它 HTTP 命令的请求头中的 key 相同，则会使用本命令中设置的 HTTP 请求头。		

8. AT BLE 指令集

8.1. AT+BLEENABLE

表 8-1. 使能 ble 模块

指令	响应
执行指令 AT+BLEENABLE	执行结果
参数： 	
示例 1： AT+BLEENABLE 正确响应： OK	

8.2. AT+BLEDISABLE

表 8-2. 失能 ble 模块

指令	响应
执行指令 AT+BLEDISABLE	执行结果
参数： 	
示例 1： AT+BLEDISABLE 正确响应： OK	

8.3. AT+BLENAME

表 8-3. 设置名称

指令	响应
帮助指令 AT+BLENAME=?	+BLENAME=<name>
查询指令 AT+BLENAME?	+BLENAME:<name>
参数： <name>：返回设备名称	

指令	响应
执行指令 AT+BLENAME=<name>	执行结果
参数： <name>：设备名称，最大长度：31，默认名称为“GD-BLE”和蓝牙地址的组合	
示例 1： AT+BLENAME? 正确响应 1： +BLENAME:GD-BLE-01:23:45:67:89:ab OK 示例 2： AT+BLENAME=test 正确响应 2： OK 备注：默认搜索到广播名为 GD-BLE+蓝牙地址，例如：GD-BLE-76:BA:ED:27:00:90	

8.4. AT+BLEADVSTART

表 8-4. 开启蓝牙广播

指令	响应
帮助指令 AT+BLEADVSTART=?	+BLEADVSTART=<type>,[intv],[ch_map],[prop],[pri_phy],[sec_phy],[wl_enable],[own_addr_type],[disc_mode],[addr_type],[addr]
执行指令 AT+BLEADVSTART=<type>,[intv],[ch_map],[prop],[pri_phy],[sec_phy],[wl_enable],[own_addr_type],[disc_mode],[addr_type],[addr]	执行结果
参数: <type>：广播类型 0: BLE_ADV_TYPE_LEGACY 1: BLE_ADV_TYPE_EXTENDED 2: BLE_ADV_TYPE_PERIODIC [intv]：广播间隔(16 进制)。参数范围：0x20 - 0x4000。广播间隔是该参数乘以 0.625 ms，所以实际的扫描窗口范围为 20ms ~ 10240ms [ch_map]：信道选择，可以按位选择信道，37,38,39 信道全部选择，该参数为 7 1: BLE_GAP_ADV_CHANN_37	

指令	响应
2: BLE_GAP_ADV_CHANN_38 4: BLE_GAP_ADV_CHANN_39 [property]：属性配置, legacy advertising 见 ble_gap_legacy_adv_prop_t, extended advertising 见 ble_gap_extended_adv_prop_t, periodic advertising 见 ble_gap_periodic_adv_prop_t [pri_phy] : primary 信道 phy 1: BLE_GAP_PHY_1MBPS 3: BLE_GAP_PHY_CODED [sec_phy]: secondary 信道 phy 1: BLE_GAP_PHY_1MBPS 2: BLE_GAP_PHY_2MBPS 3: BLE_GAP_PHY_CODED [wl_enable] : white list 是否使能 false：不使能 true：使能 [own_addr_type]：本地地址类型 0: BLE_GAP_LOCAL_ADDR_STATIC 1: BLE_GAP_LOCAL_ADDR_RESOLVABLE 2: BLE_GAP_LOCAL_ADDR_NONE_RESOLVABLE [disc_mode]：发现模式 0: BLE_GAP_ADV_MODE_NON_DISC 1: BLE_GAP_ADV_MODE_GEN_DISC 2: BLE_GAP_ADV_MODE_LIM_DISC 3: BLE_GAP_ADV_MODE_BEACON [addr_type]：对端地址类型 0: BLE_GAP_ADDR_TYPE_PUBLIC 1: BLE_GAP_ADDR_TYPE_RANDOM [addr]：对端地址	
示例 1： AT+BLEADVSTART=0 正确响应 1： OK 示例 2： AT+BLEADVSTART=0,a0,7,3,1,1,0,0,1,1F 正确响应 1： OK 备注：默认搜索到广播名为 GD-BLE+蓝牙地址，例如：GD-BLE-76:BA:ED:27:00:90	

8.5. AT+BLEADVSTOP

表 8-5. 停止蓝牙广播

指令	响应
执行指令 AT+BLEADVSTOP	执行结果
参数： 	
示例 1： AT+BLEADVSTOP 正确响应 1： OK	

8.6. AT+BLEADVDATA

表 8-6. 设置广播内容

指令	响应
帮助指令 AT+BLEADVDATA=?	+BLEADVDATA=<data>
执行指令 AT+BLEADVDATA=<data>	执行结果
参数： <data> ：广播内容，为 hex 字符串，例如 AT+BLEADVDATA=” 020106020941”代表将广播数据设置为” 0x02 0x01 0x06 0x02 0x09 0x41”	
示例 1： AT+BLEADVDATA="020106020941" 正确响应 1： OK	

8.7. AT+BLEADVDATAEX

表 8-7. 设置广播内容

指令	响应
帮助指令 AT+BLEADVDATAEX=?	+BLEADVDATAEX =<dev_name>,<uuid>,<manufacturer_data>,<include_power>
执行指令	执行结果

指令	响应
AT+BLEADVDATAEX =<dev_name>,<uuid>,<manufacturer_data>,<include_power>	
参数： <dev_name>: 设备名称 <uuid>: service uuid <manufacturer_data>: 厂家数据 <include_power>: 是否包含 tx power 1: 包含 tx 0: 不包含 tx	
示例 1： AT+BLEADVDATAEX="test","a002","2b0c112233",1 正确响应 1： OK	

8.8. AT+BLESCANRSPDATA

表 8-8. 设置扫描回复内容

指令	响应
帮助指令 AT+BLESCANRSPDATA=?	+BLESCANRSPDATA=<data>
执行指令 AT+BLESCANRSPDATA=<data>	执行结果
参数： <data> ：广播内容，为 hex 字符串，例如 AT+BLESCANRSPDATA=” 020941”代表将广播数据设置为”0x02 0x09 0x41”	
示例 1： AT+BLESCANRSPDATA="020941" 正确响应 1： OK	

8.9. AT+BLEPASSTH

表 8-9. 开启透传模式

指令	响应
帮助指令 AT+BLEPASSTH=?	+BLEPASSTH =<conn_idx>

指令	响应
执行指令 AT+BLEPASSTH=<conn_idx>	执行结果
参数： <conn_idx>: 连接索引	
示例 1： 先建立连接 AT+BLECONN=0,<addr>或者先开启广播 AT+BLEADVSTART=0，对端建立连线 输入 AT+BLEPASSTH=0 开启透传模式 “+++”退出 无响应	

8.10. AT+BLEPASSTHCLI

表 8-10. 开启透传模式

指令	响应
帮助指令 AT+BLEPASSTHCLI=?	+ BLEPASSTHCLI=<conn_idx>
执行指令 AT+BLEPASSTHCLI=<conn_idx>	执行结果
参数： <conn_idx>: 连接索引	
示例 1： AT+BLECONN=0, AB:89:67:45:23:01（对端地址），对端与之建立连线 输入 AT+BLEPASSTHCLI=0 开启透传模式 “+++”退出 无响应	

8.11. AT+BLEPASSTHAUTO

表 8-11. 自动开启透传模式

指令	响应
帮助指令 AT+BLEPASSTHAUTO=?	+BLEPASSTHAUTO=<enable>
执行指令 AT+BLEPASSTHAUTO=<e	执行结果

指令	响应
nable>	
参数： <enable> ：是否开启自动进入透传模式 1: 开启自动进入透传模式功能 0: 关闭自动进入透传模式功能	
注:主从都通过同一命令自动进入透传模式 示例 1： 输入 AT+BLEPASSTHAUTO=1 先开启广播 AT+BLEADVSTART=0，或者 AT+BLECONN=0, AB:89:67:45:23:01（对端地址）,建立连接后自动开启透传模式 “+++”退出 正确响应 1： OK	

8.12. AT+BLESCANPARAM

表 8-12. 设置扫描参数

指令	响应
帮助指令 AT+BLESCANPARAM=?	+BLESCANPARAM=<type>,<own_addr_type>,<dup_filt_pol>,<scan_intv_1m>,<scan_win_1m>,<scan_intv_coded>,<scan_ win_ coded >
查询指令 AT+BLESCANPARAM?	+BLESCANPARAM:<type>,<own_addr_type>,<dup_filt_pol>,<scan_intv_1m>,<scan_win_1m>,<scan_intv_ coded >,<scan_win_ coded >
参数： <type> ：扫描类型 0: BLE_GAP_SCAN_TYPE_GEN_DISC 1: BLE_GAP_SCAN_TYPE_LIM_DISC 2: BLE_GAP_SCAN_TYPE_OBSERVER 3: BLE_GAP_SCAN_TYPE_SEL_OBSERVER 4: BLE_GAP_SCAN_TYPE_CONN_DISC 5: BLE_GAP_SCAN_TYPE_SEL_CONN_DISC <own_addr_type> ：本地地址类型 0: BLE_GAP_LOCAL_ADDR_STATIC 1: BLE_GAP_LOCAL_ADDR_RESOLVABLE 2: BLE_GAP_LOCAL_ADDR_NONE_RESOLVABLE <dup_filt_pol> ：重复包过滤政策 0: BLE_GAP_DUP_FILT_DIS 1: BLE_GAP_DUP_FILT_EN 2: BLE_GAP_DUP_FILT_EN_PERIOD	

指令	响应
<scan_intv_1m> (16 进制)： 1M 扫描间隔。参数值应大于等于<scan_win_1m> 参数值。参数范围： 0x4-0x4000。扫描间隔是该参数乘以 0.625 ms，所以实际的扫描间隔范围为 2.5ms-10240ms。 <scan_win_1m>(16 进制) ： 1M 扫描窗口。本参数值应小于等于<scan_intv_1m> 参数值。参数范围： 0x4-0x4000。扫描窗口是该参数乘以0.625 ms，所以实际的扫描窗口范围为2.5ms-10240ms。 <scan_intv_coded>(16 进制) ： coded 扫描间隔。参数值应大于等于<scan_win_ coded> 参数值。参数范围： 0x4-0x4000。扫描间隔是该参数乘以 0.625 ms，所以实际的扫描间隔范围为 2.5ms-10240ms。 <scan_win_ coded>(16 进制) ： 1M 扫描窗口。本参数值应小于等于<scan_intv_coded> 参数值。参数范围： 0x4-0x4000。扫描窗口是该参数乘以0.625 ms，所以实际的扫描窗口范围为2.5ms-10240ms。	
执行指令 AT+BLESCANPARAM:<type>,<own_addr_type>,<dup_filt_pol>,<scan_intv_1m>,<scan_win_1m>,<scan_intv_coded>,<scan_win_coded>	执行结果
参数： <type> ：扫描类型 0: BLE_GAP_SCAN_TYPE_GEN_DISC 1: BLE_GAP_SCAN_TYPE_LIM_DISC 2: BLE_GAP_SCAN_TYPE_OBSERVER 3: BLE_GAP_SCAN_TYPE_SEL_OBSERVER 4: BLE_GAP_SCAN_TYPE_CONN_DISC 5: BLE_GAP_SCAN_TYPE_SEL_CONN_DISC <own_addr_type> ：本地地址类型 0: BLE_GAP_LOCAL_ADDR_STATIC 1: BLE_GAP_LOCAL_ADDR_RESOLVABLE 2: BLE_GAP_LOCAL_ADDR_NONE_RESOLVABLE <dup_filt_pol> ：重复包过滤政策 0: BLE_GAP_DUP_FILT_DIS 1: BLE_GAP_DUP_FILT_EN 2: BLE_GAP_DUP_FILT_EN_PERIOD <scan_intv_1m > (16 进制)： 1M 扫描间隔。参数值应大于等于< scan_win_1m > 参数值。参数范围： 0x4-0x4000。扫描间隔是该参数乘以 0.625 ms，所以实际的扫描间隔范围为 2.5ms-10240ms。 <scan_win_1m> (16 进制)： 1M 扫描窗口。本参数值应小于等于< scan_intv_1m l> 参数值。参数范围： 0x4-0x4000。扫描窗口是该参数乘以0.625 ms，所以实际的扫描窗口范围为2.5ms-10240ms。 <scan_intv_coded > (16 进制)： coded 扫描间隔。参数值应大于等于< scan_win_ coded > 参数值。参数范围：	

指令	响应
0x4-0x4000。扫描间隔是该参数乘以 0.625 ms，所以实际的扫描间隔范围为 2.5ms-10240ms。 <scan_win_ coded > (16 进制)： 1M 扫描窗口。本参数值应小于等于< scan_intv_coded > 参数值。参数范围： 0x4-0x4000。扫描窗口是该参数乘以 0.625 ms，所以实际的扫描窗口范围为 2.5ms-10240ms。	
示例 1： AT+BLESCANPARAM? 正确响应 1： +BLESCANPARAM:0,0,1,a0,20,a0,20 OK 示例 2： AT+BLESCANPARAM=0,0,1,a0,30,a0,30 正确响应 2： OK	

8.13. AT+BLESCAN

表 8-13. 开启扫描

指令	响应
帮助指令 AT+BLESCAN=?	+BLESCAN=<enable>
执行指令 AT+BLESCAN=<enable>	执行结果
参数： <type> ：扫描类型 0: 停止扫描 1: 开启扫描	
示例 1： AT+BLESCAN=1 正确响应 1： OK SCAN 结果上报： +BLESCAN: E0:48:24:7B:12:19,0,-72,0, +BLESCAN: 78:E5:B1:0C:70:7A,1,-66,1, +BLESCAN: 45:9F:72:E8:FD:DD,1,-66,2, +BLESCAN: 62:B2:5C:41:A5:D6,1,-42,3, +BLESCAN: 7C:40:88:45:B2:A4,1,-62,4, +BLESCAN: AD:25:10:14:14:D0,0,-94,5, +BLESCAN: 65:BA:4A:A4:C9:0F,1,-90,6,	

指令	响应
+BLESCAN: 8C:EA:48:73:E6:73,0,-94,7, +BLESCAN: 45:98:F5:62:26:DD,1,-59,8, +BLESCAN: 4C:08:0A:75:AE:83,1,-49,9, +BLESCAN: 61:D4:AC:00:30:0C,1,-75,10, +BLESCAN: 52:70:EC:4B:2C:E8,1,-48,11, +BLESCAN: 8C:EA:48:B7:69:C9,0,-65,12, +BLESCAN: 7C:FA:80:0C:3B:6D,0,-52,13, +BLESCAN: 44:C0:01:10:F6:42,1,-50,14, +BLESCAN: C7:5C:6F:D2:79:9F,1,-90,15,BYD BLE3 参数： +BLESCAN: <addr>, <addr type>,<rssi>,<dev idx> ,<name > <addr>: 扫描到设备的地址 <addr type>:扫描到设备的地址的地址类型 <rssi>:扫描到设备的 rssi <dev idx>:扫描到设备的索引 <name >:扫描到设备的名称	

8.14. AT+BLECONN

表 8-14. BLE 建立连接

指令	响应
帮助指令 AT+BLECONN=?	+BLECONN=<addr_type>,<addr>
执行指令 AT+BLECONN=<addr_type >, <addr>	执行结果
参数： < addr_type > ：对端地址类型 0: BLE_GAP_ADDR_TYPE_PUBLIC 1: BLE_GAP_ADDR_TYPE_RANDOM <addr> ：对端地址	
示例 1： 对端先开启广播 AT+BLECONN=0, AB:89:67:45:23:01（对端地址） 正确响应 1： OK 备注：成功连接上报：+BLECONN:<conn_idx>,<addr_type>,<addr>	

指令	响应
conn_idx 为连接索引 addr_type 为地址类型<addr>为地址	

8.15. AT+BLESCONNPARAM

表 8-15. 设置/查询连接参数

指令	响应
帮助指令 AT+BLECONNPARAM=?	+BLECONNPARM=<conn_idx>,<interval>,<latancy>,<supv_to>
查询指令 AT+BLECONNPARAM?	+BLECONNPARAM:<conn_idx>,<interval>,<latancy>,<supv_to>
参数： <conn_idx>: 连接索引 < interval > (16 进制): 连接间隔，参数范围： 0x0006-0x0C80。连接间隔等于该参数乘以 1.25 ms，所以实际的最小连接间隔范围为 7.5-4000ms。 <latancy>(16 进制): 延迟。参数范围： 0x0000-0x01F3 <supv_to>(16 进制): 超时。参数范围 0x000A-0x0C80。超时等于该参数乘以 10 ms，所以实际的超时 范围为100-32000ms。	
执行指令 AT+BLECONNPARAM=<conn_idx>,<interval>,<latancy>,<supv_to>	执行结果
参数： <conn_idx>: 连接索引 < interval >(16 进制) : 连接间隔，参数范围： 0x0006-0x0C80。连接间隔等于该参数乘以 1.25 毫秒，所以实际的最小连接间隔范围为 7.5-4000ms。 <latancy>(16 进制): 延迟。参数范围： 0x0000-0x01F3 <supv_to>(16 进制): 超时。参数范围 0x000A-0x0C80。超时等于该参数乘以 10 ms，所以实际的超时 范围为 100-32000ms。	
示例 1：查询连接参数 先建立连接 AT+BLECONN=0,<addr> 或者先开启广播 AT+BLEADVSTART=0，对端建立连线 输入 AT+BLECONNPARAM? 正确响应 1： +BLECONNPARM:0,28,0,1f4 OK 示例 2：修改连接参数	

指令	响应
先建立连接 AT+BLECONN=0,<addr>或者先开启广播 AT+BLEADVSTART=0，对端建立连线 输入 AT+BLECONNPARAM=0,40,0e,1ee 正确响应 2： OK	

8.16. AT+BLEDISCONN

表 8-16. BLE 断开连接

指令	响应
帮助指令 AT+BLEDISCONN=?	+BLEDISCONN=<conn_idx>
执行指令 AT+BLEDISCONN=<conn_idx>	执行结果
参数： <conn_idx>: 连接索引	
示例 1： 先建立连接 AT+BLECONN=0,<addr>或者先开启广播 AT+BLEADVSTART=0，对端建立连线 输入 AT+BLEDISCONN=0 正确响应 1： OK 备注：成功连接上报：+BLEDISCONN:< conn_idx >,<reseon> conn_idx 为连接索引，reseon 为断线原因	

8.17. AT+BLEMTU

表 8-17. 更新/查询 mtu

指令	响应
帮助指令 AT+BLEMTU=?	+BLEMTU=<conn_idx>,<pref_mtu>
查询指令 AT+BLEMTU?	+BLEMTU:<conn_idx>,<mtu_size>

指令	响应
参数： <conn_idx>: 连接索引 <mtu_size>: 现有mtu值	
执行指令 AT+BLEMTU=<conn_idx>, <pref_mtu>	执行结果
参数： <conn_idx>: 连接索引 <pref_mtu>: 期望 mtu	
示例 1：查询 mtu 先建立连接 AT+BLECONN=0,<addr>或者先开启广播 AT+BLEADVSTART=0，对端建立连线 输入 AT+BLEMTU? 正确响应 1： +BLEMTU:0,23 OK 示例 2：更新 MTU 先建立连接 AT+BLECONN=0,<addr> 输入 AT+BLEMTU=0,1000 正确响应 2： OK	

8.18. AT+BLEPHY

表 8-18. 更新/查询 phy

指令	响应
帮助指令 AT+BLEPHY=?	+BLEPHY=<conn_idx>,<tx_phy>,<rx_phy>,<phy_opt>
查询指令 AT+BLEPHY?	+BLEPHY:<conn_idx>,<tx_phy>,<rx_phy>
参数： <conn_idx>: 连接索引 <tx_phy>: tx 用到的 phy，0 代表 no preferred PHY，详见 ble_gap_phy_bf_t <rx_phy>: tx 用到的 phy，0 代表 no preferred PHY，详见 ble_gap_phy_bf_t <phy_opt>:coded phy 500k， 125k 选项，0 代表 no preference rate for coded phy 详见 ble_gap_phy_option	
执行指令 AT+BLEPHY=<conn_idx>,<	执行结果

指令	响应
tx_phy>,<rx_phy>,<phy_opt>	
参数： <conn_idx>: 连接索引 <tx_phy>: tx 用到的 phy，0 代表 no preferred PHY，详见 ble_gap_phy_bf_t <rx_phy>: tx 用到的 phy，0 代表 no preferred PHY，详见 ble_gap_phy_bf_t <phy_opt>:coded phy 500k， 125k 选项，0 代表 no preference rate for coded phy 详见 ble_gap_phy_option	
示例 1：查询 phy 先建立连接 AT+BLECONN=0,<addr>或者先开启广播 AT+BLEADVSTART=0，对端建立连线 输入 AT+BLEPHY? 正确响应 1： +BLEPHY:0,0,0 OK 示例 2：更新 phy 先建立连接 AT+BLECONN=0,<addr>或者先开启广播 AT+BLEADVSTART=0，对端建立连线 输入 AT+BLEPHY=0,1,1,0 正确响应 2： OK	

8.19. AT+BLEDATALEN

表 8-19. Data length extension

指令	响应
帮助指令 AT+BLEDATALEN=?	+BLEDATALEN=<conn_idx>,<tx_oct>
执行指令 AT+BLEDATALEN=<conn_idx>,<tx_oct>	执行结果
参数： <conn_idx>: 连接索引 <tx_oct>:期望的 le 数据包长度，范围： 0x001B-0x00FB	
示例 1： 先建立连接 AT+BLECONN=0,<addr>或者先开启广播 AT+BLEADVSTART=0，对端建立连线 输入 AT+BLEDATALEN=0,200 正确响应 1： OK	

8.20. AT+BLEADDR

表 8-20. 查询/设置 ble bd address

指令	响应
帮助指令 AT+BLEADDR=?	+BLEADDR=<bd_addr>
查询指令 AT+BLEADDR?	+BLEADDR:<bd_addr>
参数： < bd_addr > ：蓝牙地址	
执行指令 AT+BLEADDR=<bd_addr>	执行结果
参数： < bd_addr > ：蓝牙地址	
示例 1： 查询 ble bd address AT+BLEADDR? 正确响应 1： +BLEBDADDR:77:66:55:44:33:22 OK 示例 2：设置 ble bd address AT+BLEADDR=22:33:44:55:66:77 正确响应 2： OK 备注：重启后生效	

8.21. AT+BLESETAUTH

表 8-21. 配置 AUTHENTICATION

指令	响应
帮助指令 AT+BLESETAUTH=?	+BLESETAUTH=<bond>,<mitm>,<sc>,<iocap>,<oob>,<key_size>

指令	响应
执行指令 AT+BLESETAUTH=<bond>,<mitm>,<sc>,<iocap>,<oob>,<key_size>	执行结果
参数： < bond > : bonding flag 0x00: no bonding 0x01: bonding <mitm>: mitm flag 0x00: mitm protection not required 0x01: mitm protection required < sc >: secure connections flag 0x00: secure connections pairing is not supported 0x01: secure connections pairing is supported < iocap>: io capability to set 0x00: display only 0x01: display yes no 0x02: keyboard only 0x03: no input no output 0x04: keyboard display <oob>: oob flag for authention 0x00: without oob 0x01: with oob [key size]: encryption key size requirement, default is 16 if not set	
示例 1：配置 AUTHENTICATION AT+BLESETAUTH=1,0,0,3,0,16 正确响应 1： OK	

8.22. AT+BLEPAIR

表 8-22. 发起配对

指令	响应
帮助指令 AT+BLEPAIR=?	+BLEPAIR=<conidx>
执行指令 AT+BLEPAIR=<conidx>	执行结果

指令	响应
参数： <conn_idx>: 连接索引	
示例 1：发起配对 先建立连接 AT+BLECONN=0,<addr>或者先开启广播 AT+BLEADVSTART=0，对端建立连线 输入 AT+BLEPAIR=0 正确响应 1： OK	

8.23. AT+BLEENCRYPT

表 8-23. 启动加密

指令	响应
帮助指令 AT+BLEENCRYPT=?	+BLEENCRYPT=<conidx >
执行指令 AT+BLEENCRYPT=<conidx >	执行结果
参数： <conn_idx>: 连接索引	
示例 1：启动加密（需要之前配对过的设备） 先建立连接 AT+BLECONN=0,<addr>或者先开启广播 AT+BLEADVSTART=0，对端建立连线 输入 AT+BLEENCRYPT=0 正确响应 1： OK	

8.24. AT+BLESETKEY

表 8-24. 设置静态配对密钥

指令	响应
帮助指令 AT+BLESETKEY=?	+BLESETKEY=<key>

指令	响应
执行指令 AT+BLESETKEY =<key>	执行结果
参数： <key>: 6 位静态配对密钥	
示例 1：设置静态配对密钥 输入 AT+ BLESETKEY =123456， 正确响应 1： OK 输入 AT+BLESETAUTH=1,1,0,1,0,16， 先建立连接 AT+BLECONN=0,<addr>或者先开启广播 AT+BLEADVSTART=0，对端建立连线 对端提示输入 PIN code 后在对端输入 123456	

8.25. AT+BLEPASSKEY

表 8-25. 输入 passkey

指令	响应
帮助指令 AT+BLEPASSKEY=?	+BLEPASSKEY=<conidx>,<passkey>
执行指令 AT+BLEPASSKEY=<conidx>,<passkey>	执行结果
参数： <conn_idx>: 连接索引 <passkey>: 6 位数字密码	
示例 1：输入 passkey 输入 AT+BLESETAUTH=1,1,0,2,0,16 先建立连接 AT+BLECONN=0,<addr>或者先开启广播 AT+BLEADVSTART=0，对端建立连线 收到+BLEPASSKEYREQ:<conn idx>上报后，输入 AT+BLEPASSKEY=<conn idx>,123456（输入显示的值） 正确响应 1： OK	

8.26. AT+BLECOMPARE

表 8-26. 输入 compare 结果

指令	响应
帮助指令 AT+BLECOMPARE=?	+BLECOMPARE=<conidx>,<value>
执行指令 AT+BLECOMPARE=<conidx>,<value>	执行结果
参数： <conn_idx>: 连接索引 <value>:比较结果, 1 表示成功， 0 表示失败	
示例 1：输入 compare 结果 输入 AT+BLESETAUTH=1,1,1,4,0,16 先建立连接 AT+BLECONN=0,<addr>或者先开启广播 AT+BLEADVSTART=0，对端建立连线 收到+BLECOMPAREREQ:<conn idx>,<number> 上报后，比对两端显示的数字，输入 AT+BLECOMPARE=<conn idx>,1 正确响应 1： OK	

8.27. AT+BLELISTENCDEV

表 8-27. 列出 bond device 列表

指令	响应
查询指令 AT+BLELISTENCDEV?	+BLELISTENCDEV:<dev_idx>,<addr>
参数： < dev_idx >: 设备索引 < addr >: 设备地址	
示例 1：列出 bond device 列表 需要先有设备配对过 AT+BLELISTENCDEV? 正确响应 1： +BLELISTENCDEV=0,AB:89:67:45:23:01 +BLELISTENCDEV=1,D0:20:DD:EE:5C:3C OK	

指令	响应

8.28. AT+BLECLEARENCDEV

表 8-28. 移除 bond 设备

指令	响应
帮助指令 AT+BLECLEARENCDEV=?	+BLECLEARENCDEV=<dev_idx>
执行指令 AT+BLECLEARENCDEV=<dev_idx>	执行结果
参数： < dev_idx >: 设备索引	
示例 1：输入移除 bond 设备 需要先有设备配对过 AT+BLECLEARENCDEV=0 正确响应 1： OK	

8.29. AT+BLEGATTSSVC

表 8-29. 列出本地注册的 service

指令	响应
查询指令 AT+BLEGATTSSVC?	+BLEGATTSSVC:<svc_id><uuid>< svc_type >
参数： < svc_id >: 服务索引 < uuid >: service uuid < svc_type >: service 类型 0x01: BLE_GATT_ATTR_PRIMARY_SVC 0x02: BLE_GATT_ATTR_SECONDARY_SVC 0x03: BLE_GATT_ATTR_INCL_SVC	
示例 1：列出本地注册的 service AT+BLEGATTSSVC?	

指令	响应
正确响应 1： +BLEGATTSSVC:0,180A,1 +BLEGATTSSVC:1,00001111000000000123456789ABCDEF,1 +BLEGATTSSVC:2,FFF0,1 +BLEGATTSSVC:3,0101,1 OK	

8.30. AT+BLEGATTSLISTALL

表 8-30. 列出本地所有 service 中信息

指令	响应
查询指令 AT+BLEGATTSLISTALL ?	+BLEGATTSSVC:<svc_id><uuid>< svc_type > +BLEGATTSCHAR: <svc_id><value_index><uuid > +BLEGATTSDESC: <svc_id><desc_idx><uuid >
参数： < svc_id >: service 索引 < uuid >: service uuid < svc_type >: service 类型 0x01: BLE_GATT_ATTR_PRIMARY_SVC 0x02: BLE_GATT_ATTR_SECONDARY_SVC 0x03: BLE_GATT_ATTR_INCL_SVC < svc_id >: service 索引 < value_index >: characteristic value index < uuid >: characteristic uuid <svc_id>: service 索引 <desc_idx>: descriptor 索引 <uuid >:descriptor uuid	
示例 1：**列出本地所有 service 中信息** AT+BLEGATTSLISTALL? 正确响应 1： +BLEGATTSSVC:0,180A,1 +BLEGATTSCHAR:,0,2,2A29 +BLEGATTSCHAR:,0,4,2A24 +BLEGATTSCHAR:,0,6,2A25 +BLEGATTSCHAR:,0,8,2A27 +BLEGATTSCHAR:,0,10,2A26 +BLEGATTSCHAR:,0,12,2A28 +BLEGATTSCHAR:,0,14,2A23 +BLEGATTSCHAR:,0,16,2A2A +BLEGATTSCHAR:,0,18,2A50 +BLEGATTSSVC:1,00001111000000000123456789ABCDEF,1 +BLEGATTSCHAR:,1,2,00002222000000000123456789ABCDEF +BLEGATTSCHAR:,1,4,00003333000000000123456789ABCDEF	

指令	响应
+BLEGATTSCHAR:,1,6,000044440000000000123456789ABCDEF +BLEGATTSDESC:,1,7,2902 +BLEGATTSSVC:2,FFF0,1 +BLEGATTSCHAR:,2,2,FFF1 +BLEGATTSCHAR:,2,4,FFF2 +BLEGATTSDESC:,2,5,2902 +BLEGATTSSVC:3,0101,1 +BLEGATTSCHAR:,3,2,0102 +BLEGATTSCHAR:,3,4,0103 +BLEGATTSDESC:,3,5,2902 OK	

8.31. AT+BLEGATTSNTF

表 8-31. 发送 notification

指令	响应
帮助指令 AT+BLEGATTSNTF=?	+BLEGATTSNTF=<conn_idx>,<svc_id>,<char_idx>,<tx_len>
执行指令 AT+BLEGATTSNTF=<conn_idx>,<svc_id>,<char_idx>,<tx_len>	执行结果
参数： <conn_idx>: 连接索引 <svc_id>: service id <char_idx>: characteristic value index <tx_len>: tx数据长度	
示例 1：发送 notification 先开启广播 AT+BLEADVSTART=0，对端建立连线 输入 AT+BLEGATTSNTF=0,1,6,5 > 输入 AAAAA(对端会收到数据) 正确响应 1： OK	

8.32. AT+BLEGATTSIND

表 8-32. 发送 indication

指令	响应
帮助指令 AT+BLEGATTSIND=?	+BLEGATTSIND=<conn_idx>,<svc_id>,<char_idx>,<tx_len>
执行指令 AT+BLEGATTSIND=<conn_idx>,<svc_id>,<char_idx>,<tx_len>	执行结果
参数： <conn_idx>: 连接索引 <svc_id>: service id <char_idx>: characteristic value index <tx_len>: tx数据长度	
示例 1：发送 indication 先开启广播 AT+BLEADVSTART=0，对端建立连线 输入 AT+BLEGATTSIND=0,1,6,5 > 输入 AAAAA(对端会收到数据) 正确响应 1： OK	

8.33. AT+BLEGATTSSETATTRVAL

表 8-33. 设置 characteristic 的值

指令	响应
帮助指令 AT+BLEGATTSSETATTRVAL=?	+BLEGATTSSETATTRVAL=<conn_idx>,<svc_id>,<char_idx>,<tx_len>
执行指令 AT+BLEGATTSSETATTRVAL=<conn_idx>,<svc_id>,<char_idx>,<tx_len>	执行结果
参数： <conn_idx>: 连接索引 <svc_id>: service id <char_idx>: characteristic value index <tx_len>: tx数据长度	

指令	响应
示例 1：设置 characteristic 的值 先开启广播 AT+BLEADVSTART=0，对端建立连线 输入 AT+BLEGATTSSETATTRVAL=0,1,4,5 > 输入 AAAAA(本地数据改变) 正确响应 1： OK	

8.34. AT+BLEGATTCDISCSVC

表 8-34. 发现 service

指令	响应
帮助指令 AT+BLEGATTCDISCSVC=?	+BLEGATTCDISCSVC=<conn_idx>,<start_hdl>,<end_hdl>
执行指令 AT+BLEGATTCDISCSVC=<conn_idx>,<start_hdl>,<end_hdl>	执行结果 +BLEGATTCDISCSVC:<start_hdl >,<end_hdl >,<uuid>
参数： 执行指令: <conn_idx>: 连接索引 < start_hdl >: attribute start handle < end_hdl >: attribute end handle 执行结果: < start_hdl >: attribute start handle < end_hdl >: attribute end handle <uuid>: 连接索引	
示例 1：发现 service 先建立连接 AT+BLECONN=0,<addr> 输入 AT+BLEGATTCDISCSVC=0,1,ffff 正确响应 1： +BLEGATTCDISCSVC:01,08,00001111000000000123456789ABCDEF +BLEGATTCDISCSVC:09,0e,FFF0 +BLEGATTCDISCSVC:10,19,1801 +BLEGATTCDISCSVC:1a,1f,0101 +BLEGATTCDISCSVC:20,28,1800 +BLEGATTCDISCSVC:2b,3d,180A OK	

8.35. AT+BLEGATTCDISCCHAR

表 8-35. 发现 characteristic

指令	响应
帮助指令 AT+BLEGATTCDISCCHAR=?	+BLEGATTCDISCCHAR=<conn_idx>,<start_hdl>,<end_hdl>
执行指令 AT+BLEGATTCDISCCHAR=<conn_idx>,<start_hdl>,<end_hdl>	执行结果 +BLEGATTCDISCCHAR:<char_hdl >,<val_hdl >,<prop >,<uuid>
参数： 执行指令: <conn_idx>: 连接索引 < start_hdl >: attribute start handle < end_hdl >: attribute end handle 执行结果: <char_hdl >: characteristic index <val_hdl >: characteristic value index <prop >: characteristic properties，参考 ble_gatt_attr_info_bf <uuid>: uuid	
示例 1：发现 characteristic 先建立连接 AT+BLECONN=0,<addr> 输入 AT+BLEGATTCDISCCHAR=0,1,ffff 正确响应 1： +BLEGATTCDISCCHAR:02,03,02,00002222000000000123456789ABCDEF +BLEGATTCDISCCHAR:04,05,0c,00003333000000000123456789ABCDEF +BLEGATTCDISCCHAR:06,07,30,00004444000000000123456789ABCDEF +BLEGATTCDISCCHAR:0a,0b,08,FFF1 +BLEGATTCDISCCHAR:0c,0d,10,FFF2 +BLEGATTCDISCCHAR:11,12,20,2A05 +BLEGATTCDISCCHAR:14,15,0a,2B29 +BLEGATTCDISCCHAR:16,17,02,2B2A +BLEGATTCDISCCHAR:18,19,02,2B3A +BLEGATTCDISCCHAR:1b,1c,0c,0102 +BLEGATTCDISCCHAR:1d,1e,10,0103 +BLEGATTCDISCCHAR:21,22,0a,2A00 +BLEGATTCDISCCHAR:23,24,0a,2A01 +BLEGATTCDISCCHAR:25,26,02,2A04 +BLEGATTCDISCCHAR:27,28,02,2AA6 +BLEGATTCDISCCHAR:2c,2d,02,2A29 +BLEGATTCDISCCHAR:2e,2f,02,2A24 +BLEGATTCDISCCHAR:30,31,02,2A25 +BLEGATTCDISCCHAR:32,33,02,2A27 +BLEGATTCDISCCHAR:34,35,02,2A26 +BLEGATTCDISCCHAR:36,37,02,2A28	

指令	响应
+BLEGATTCDISCCHAR:38,39,02,2A23 +BLEGATTCDISCCHAR:3a,3b,02,2A2A +BLEGATTCDISCCHAR:3c,3d,02,2A50 OK	

8.36. AT+BLEGATTCDISCDESC

表 8-36. 发现 descriptor

指令	响应
帮助指令 AT+BLEGATTCDISCDESC=?	+BLEGATTCDISCDESC=<conn_idx>,<start_hdl>,<end_hdl>
执行指令 AT+BLEGATTCDISCDESC=<conn_idx>,<start_hdl>,<end_hdl>	执行结果 +BLEGATTCDISCDESC:<desc_hdl >,<uuid>
参数： 执行指令: <conn_idx>: 连接索引 < start_hdl >: attribute start handle < end_hdl >: attribute end handle 执行结果: < desc_hdl >: descriptor index <uuid>: uuid	
示例 1：发现 descriptor 先建立连接 AT+BLECONN=0,<addr> 输入 AT+BLEGATTCDISCDESC=0,1,ffff 正确响应 1： +BLEGATTCDISCDESC:08,2902 +BLEGATTCDISCDESC:0e,2902 +BLEGATTCDISCDESC:13,2902 +BLEGATTCDISCDESC:1f,2902 OK	

8.37. AT+BLEGATTCRD

表 8-37. Read attribute value

指令	响应
帮助指令 AT+BLEGATTCRD=?	+BLEGATTCRD=<conn_idx>,<handle>,<max_len>

指令	响应
执行指令 AT+BLEGATTCRD=<conn_idx>,<handle>,<max_len>	执行结果 +BLEGATTCRD:<conn_idx >,<length>,<data>
参数： 执行指令: <conn_idx>: 连接索引 < handle >: attribute handle < max_len >: read 最大长度 执行结果: <conn_idx >: 连接索引 <length>: 读出数据长度 <data>: 数据内容	
示例 1：Read attribute value 先建立连接 AT+BLECONN=0,<addr> 输入 AT+BLEGATTCRD=0,3,100 正确响应 1： +BLEGATTCRD:0,2, AA OK	

8.38. AT+BLEGATTCWR

表 8-38. Write attribute value

指令	响应
帮助指令 AT+BLEGATTCWR=?	+BLEGATTCWR=<conn_idx>,<handle>,<write_type>,<len>
执行指令 AT+BLEGATTCWR=<conn_idx>,<handle>,<write_type>,<len>	执行结果
参数： 执行指令: <conn_idx>: 连接索引 < handle >: attribute handle < write_type >: 写类型 00 : BLE_GATT_WRITE 01: BLE_GATT_WRITE_NO_RESP <len>: 写长度	
示例 1：Write attribute value 先建立连接 AT+BLECONN=0,<addr>	

指令	响应
输入 AT+BLEGATTCWR=0,5,0,5 > 输入 AAAAA(对端会收到数据) 正确响应 1： OK	

8.39. AT+BLEDADATRANS

表 8-39. 进入普通传输模式

指令	响应
帮助指令 AT+ BLEDADATRANS=?	+BLEDADATRANS=<enable>
执行指令 AT+BLEDADATRANS=<enable>	执行结果
参数： 执行指令: <enable>: 开启或结束	
示例 1：进入普通传输模式 先建立连接：开启广播 AT+BLEADVSTART=0，对端建立连线 AT+BLEDADATRANS=1 正确响应 1： OK	

8.40. AT+BLEDADATRANSSEND

表 8-40. 单条发送数据（非透传）

指令	响应
帮助指令 AT+BLEDADATRANSSEND=?	+BLEDADATRANSSEND=<conn_idx>,<tx_len>
执行指令 AT+BLEDADATRANSSEND=<conn_idx>,<tx_len>	执行结果

指令	响应
参数： <conn_idx>: 连接索引 <tx_len>: tx数据长度	
示例 1：**单条发送数据（非透传）** 先开启广播 AT+BLEADVSTART=0，对端建立连线 输入 AT+BLEDADATRANSSEND=0,5 > 输入 AAAAA(对端会收到数据) 正确响应 1： OK 备注：收到对端数据：+BLEDATA:< conn_idx>,<len>,<data> conn_idx 为连接索引 len 为数据长度< data >为数据内容	

8.41. AT+BLECOURIER

表 8-41. 自动开启透传模式

指令	响应
帮助指令 AT+BLECOURIER=?	+BLEPASSTHAUTO=<enable>
执行指令 AT+BLECOURIER=<enable>	执行结果
参数： <enable> ：是否开启配网广播，进入配网模式 1: 开启启配网广播，进入配网模式 0: 关闭启配网广播，退出配网模式	
示例 1： 输入 AT+BLECOURIER =1 正确响应 1： OK	

9. 版本历史

表 9-1. 版本历史

版本号.	说明	日期
1.0	首次发布	2023 年 10 月 17 日
1.1	新增了 AT+TRANSINTVL 和 AT+CIPMODE 命令，并扩展了原有的 AT+CIPSEND 支持数据透传	2024 年 7 月 16 日
1.2	新增 BLE 相关 AT 命令	2024 年 10 月 8 日
1.3	新增 BLE 相关 AT 命令	2025 年 3 月 20 日
1.4	新增了 MQTT 和 HTTP 相关 AT 命令，并扩展了基础指令和 TCPIP 相关 AT 命令	2025 年 11 月 19 日
1.5	新增 BLE 相关 AT 命令，并且对之前的命令进行规整	2026 年 4 月 15 日

Important Notice

This document is the property of GigaDevice Semiconductor Inc. and its subsidiaries (the "Company"). This document, including any product of the Company described in this document (the “Product”), is owned by the Company according to the laws of the People’s Republic of China and other applicable laws. The Company reserves all rights under such laws and no Intellectual Property Rights are transferred (either wholly or partially) or licensed by the Company (either expressly or impliedly) herein. The names and brands of third party referred thereto (if any) are the property of their respective owner and referred to for identification purposes only.

To the maximum extent permitted by applicable law, the Company makes no representations or warranties of any kind, express or implied, with regard to the merchantability and the fitness for a particular purpose of the Product, nor does the Company assume any liability arising out of the application or use of any Product. Any information provided in this document is provided only for reference purposes. It is the sole responsibility of the user of this document to determine whether the Product is suitable and fit for its applications and products planned, and properly design, program, and test the functionality and safety of its applications and products planned using the Product. The Product is designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only, and the Product is not designed or intended for use in (i) safety critical applications such as weapons systems, nuclear facilities, atomic energy controller, combustion controller, aeronautic or aerospace applications, traffic signal instruments, pollution control or hazardous substance management; (ii) life-support systems, other medical equipment or systems (including life support equipment and surgical implants); (iii) automotive applications or environments, including but not limited to applications for active and passive safety of automobiles (regardless of front market or aftermarket), for example, EPS, braking, ADAS (camera/fusion), EMS, TCU, BMS, BSG, TPMS, Airbag, Suspension, DMS, ICMS, Domain, ESC, DCDC, e-clutch, advanced-lighting, etc.. Automobile herein means a vehicle propelled by a self-contained motor, engine or the like, such as, without limitation, cars, trucks, motorcycles, electric cars, and other transportation devices; and/or (iv) other uses where the failure of the device or the Product can reasonably be expected to result in personal injury, death, or severe property or environmental damage (collectively "Unintended Uses"). Customers shall take any and all actions to ensure the Product meets the applicable laws and regulations. The Company is not liable for, in whole or in part, and customers shall hereby release the Company as well as its suppliers and/or distributors from, any claim, damage, or other liability arising from or related to all Unintended Uses of the Product. Customers shall indemnify and hold the Company, and its officers, employees, subsidiaries, affiliates as well as its suppliers and/or distributors harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of the Product.

Information in this document is provided solely in connection with the Product. The Company reserves the right to make changes, corrections, modifications or improvements to this document and the Product described herein at any time without notice. The Company shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. Information in this document supersedes and replaces information previously supplied in any prior versions of this document.