

ESP32-C6

ESP-AT 用户指南



Release release-v4.2.0.0

乐鑫信息科技

2026 年 08 月 07 日

目录

目录	i
1 入门指南	3
1.1 ESP-AT 是什么	3
1.2 技术选型	4
1.2.1 硬件选型	4
1.2.2 AT 软件方案选型	4
1.2.3 项目初期准备	6
1.3 硬件连接	6
1.3.1 硬件准备	6
1.3.2 ESP32-C6-4MB 系列	7
1.4 下载指导	7
1.4.1 下载 AT 固件	8
1.4.2 烧录 AT 固件至设备	9
1.4.3 检查 AT 固件是否烧录成功	11
2 AT 固件	15
2.1 发布的固件	15
2.1.1 介绍	15
2.1.2 ESP32-C6-4MB 系列	15
2.1.3 订阅 AT 版本发布	16
2.1.4 AT 发布固件差异	16
2.2 AT 固件简介	17
3 AT 命令集	19
3.1 基础 AT 命令集	19
3.1.1 介绍	20
3.1.2 AT: 测试 AT 启动	20
3.1.3 AT+RST: 重启模块	20
3.1.4 AT+GMR: 查看版本信息	20
3.1.5 AT+CMD: 查询当前固件支持的所有命令及命令类型	21
3.1.6 AT+GSLP: 进入 Deep-sleep 模式	21
3.1.7 ATE: 开启或关闭 AT 回显功能	22
3.1.8 AT+RESTORE: 恢复出厂设置	22
3.1.9 AT+SAVETRANSLINK: 设置开机 Network/Bluetooth LE 透传模式信息	23
3.1.10 AT+TRANSINTVL: 设置透传模式模式下的数据发送间隔	26
3.1.11 AT+UART_CUR: 设置 UART 当前临时配置, 不保存到 flash	27
3.1.12 AT+UART_DEF: 设置 UART 默认配置, 保存到 flash	28
3.1.13 AT+SLEEP: 设置睡眠模式	29
3.1.14 AT+SYSRAM: 查询堆空间使用情况	30
3.1.15 AT+SYSMSG: 查询/设置系统提示信息	31
3.1.16 AT+SYSMSGFILTER: 启用或禁用系统消息过滤	33
3.1.17 AT+SYSMSGFILTERCFG: 查询/配置系统消息的过滤器	34
3.1.18 AT+SYSFLASH: 查询或读写 flash 用户分区	37
3.1.19 AT+SYSMFG: 查询或读写 manufacturing nvs 用户分区	38
3.1.20 AT+RFPOWER: 查询/设置 RF TX Power	41
3.1.21 AT+RFCAL: RF 全面校准	42

3.1.22	AT+SYSROLLBACK: 回滚到以前的固件	43
3.1.23	AT+SYSTIMESTAMP: 查询/设置本地时间戳	43
3.1.24	AT+SYSLOG: 启用或禁用 AT 错误代码提示	44
3.1.25	AT+SLEEPWKCFG: 设置 Light-sleep 唤醒源和唤醒 GPIO	46
3.1.26	AT+SYSSTORE: 设置参数存储模式	47
3.1.27	AT+SYSREG: 读写寄存器	48
3.2	Wi-Fi AT 命令集	48
3.2.1	介绍	49
3.2.2	AT+CWINIT: 初始化/清理 Wi-Fi 驱动程序	49
3.2.3	AT+CWMODE: 查询/设置 Wi-Fi 模式 (Station/SoftAP/Station+SoftAP)	50
3.2.4	AT+CWBANDWIDTH: 查询/设置 Wi-Fi 带宽	51
3.2.5	AT+CWSTATE: 查询 Wi-Fi 状态和 Wi-Fi 信息	52
3.2.6	AT+CWCONFIG: 查询/设置 Wi-Fi 非活动时间和监听间隔时间	53
3.2.7	AT+CWJAP: 连接 AP	54
3.2.8	AT+CWRECONNCFG: 查询/设置 Wi-Fi 重连配置	56
3.2.9	AT+CWLAPOPT: 设置 AT+CWLAP 命令扫描结果的属性	57
3.2.10	AT+CWLAP: 扫描当前可用的 AP	58
3.2.11	AT+CWQAP: 断开与 AP 的连接	60
3.2.12	AT+CWSAP: 配置 ESP32-C6 SoftAP 参数	60
3.2.13	AT+CWLIF: 查询连接到 ESP32-C6 SoftAP 的 station 信息	61
3.2.14	AT+CWQIF: 断开 station 与 ESP32-C6 SoftAP 的连接	62
3.2.15	AT+CWDHCP: 启用或禁用 DHCP	63
3.2.16	AT+CWDHCPS: 查询/设置 ESP32-C6 SoftAP DHCP 分配的 IPv4 地址范围	64
3.2.17	AT+CWAUTOCONN: 查询/设置上电是否自动连接 AP	65
3.2.18	AT+CWAPPROTO: 查询/设置 SoftAP 模式下 Wi-Fi 协议标准	65
3.2.19	AT+CWSTAPROTO: 查询/设置 Station 模式下 Wi-Fi 协议标准	66
3.2.20	AT+CIPSTAMAC: 查询/设置 ESP32-C6 Station 的 MAC 地址	67
3.2.21	AT+CIPAPMAC: 查询/设置 ESP32-C6 SoftAP 的 MAC 地址	68
3.2.22	AT+CIPSTA: 查询/设置 ESP32-C6 Station 的 IP 地址	69
3.2.23	AT+CIPAP: 查询/设置 ESP32-C6 SoftAP 的 IP 地址	70
3.2.24	AT+CWSTARTSMART: 开启 SmartConfig	71
3.2.25	AT+CWSTOPSMART: 停止 SmartConfig	72
3.2.26	AT+WPS: 设置 WPS 功能	72
3.2.27	AT+CWJEAP: 连接 WPA2/WPA3 企业版 AP	73
3.2.28	AT+CWHOSTNAME: 查询/设置 ESP32-C6 Station 的主机名称	76
3.2.29	AT+CWCOUNTRY: 查询/设置 Wi-Fi 国家代码	77
3.3	TCP/IP AT 命令	78
3.3.1	介绍	79
3.3.2	AT+CIPV6: 启用或禁用 IPv6 网络 (IPv6)	79
3.3.3	AT+CIPSTATE: 查询 TCP/UDP/SSL 连接信息	80
3.3.4	AT+CIPDOMAIN: 域名解析	80
3.3.5	AT+CIPSTART: 建立 TCP 连接、UDP 传输或 SSL 连接	81
3.3.6	AT+CIPSTARTEX: 建立自动分配 ID 的 TCP 连接、UDP 传输或 SSL 连接	86
3.3.7	[仅适用数据模式] +++: 退出数据模式	86
3.3.8	AT+CIPSEND: 在普通传输模式或 Network 透传模式下发送数据	86
3.3.9	AT+CIPSENDL: 在普通传输模式下并行发送长数据	87
3.3.10	AT+CIPSENDCFG: 设置 AT+CIPSENDL 命令的属性	88
3.3.11	AT+CIPSENDEX: 在普通传输模式下采用扩展的方式发送数据	89
3.3.12	AT+CIPCLOSE: 关闭 TCP/UDP/SSL 连接	90
3.3.13	AT+CIPCONNPERST: 查询、设置 TCP/SSL 连接持久化参数	91
3.3.14	AT+CIFSR: 查询本地 IP 地址和 MAC 地址	92
3.3.15	AT+CIPMUX: 启用或禁用多连接模式	92
3.3.16	AT+CIPSERVER: 建立/关闭 TCP 或 SSL 服务器	93
3.3.17	AT+CIPSERVERMAXCONN: 查询/设置服务器允许建立的最大连接数	95
3.3.18	AT+CIPMODE: 查询/设置传输模式	96
3.3.19	AT+CIPSTO: 查询/设置本地 TCP/SSL 服务器超时时间	97
3.3.20	AT+CIPSNTPCFG: 查询/设置时区和 SNTP 服务器	97
3.3.21	AT+CIPSNTPTIME: 查询 SNTP 时间	99

3.3.22	AT+CIPSNTPINTV: 查询/设置 SNTP 时间同步的间隔	99
3.3.23	AT+CIPFWVER: 查询服务器已有的 AT 固件版本	100
3.3.24	AT+CIUPDATE: 通过 Network 升级固件	101
3.3.25	AT+CIPDINFO: 设置 +IPD 消息详情	103
3.3.26	AT+CIPSSLCCONF: 查询/设置 SSL 客户端配置	103
3.3.27	AT+CIPSSLCCIPHER: 查询/设置 SSL 客户端的密码套件 (Cipher Suite)	104
3.3.28	AT+CIPSSLCCN: 查询/设置 SSL 客户端的公用名 (common name)	105
3.3.29	AT+CIPSSLCSNI: 查询/设置 SSL 客户端的 SNI	106
3.3.30	AT+CIPSSLCALPN: 查询/设置 SSL 客户端 ALPN	107
3.3.31	AT+CIPSSLCPSK: 查询/设置 SSL 客户端的 PSK (字符串格式)	108
3.3.32	AT+CIPSSLCPSKHEX: 查询/设置 SSL 客户端的 PSK (十六进制格式)	109
3.3.33	AT+CIPRECONNINTV: 查询/设置 Network 透传模式下的 TCP/UDP/SSL 重连间隔	109
3.3.34	AT+CIPRECVMODE: 查询/设置套接字接收模式	110
3.3.35	AT+CIPRECVDATA: 获取被动接收模式下的套接字数据	111
3.3.36	AT+CIPRECVLEN: 查询被动接收模式下套接字数据的长度	112
3.3.37	AT+PING: ping 对端主机	113
3.3.38	AT+CIPDNS: 查询/设置 DNS 服务器信息	113
3.3.39	AT+MDNS: 设置 mDNS 功能	115
3.3.40	AT+CIPTCPOPT: 查询/设置套接字选项	117
3.4	Bluetooth® Low Energy AT 命令集	118
3.4.1	介绍	119
3.4.2	AT+BLEINIT: Bluetooth LE 初始化	119
3.4.3	AT+BLEADDR: 设置 Bluetooth LE 设备地址	121
3.4.4	AT+BLENAME: 查询/设置 Bluetooth LE 设备名称	122
3.4.5	AT+BLESCANPARAM: 查询/设置 Bluetooth LE 扫描参数	123
3.4.6	AT+BLESCAN: 使能 Bluetooth LE 扫描	124
3.4.7	AT+BLESCANRSPDATA: 设置 Bluetooth LE 扫描响应	125
3.4.8	AT+BLEADVPARAM: 查询/设置 Bluetooth LE 广播参数	126
3.4.9	AT+BLEADVDATA: 设置 Bluetooth LE 广播数据	127
3.4.10	AT+BLEADVDATAEX: 自动设置 Bluetooth LE 广播数据	128
3.4.11	AT+BLEADVSTART: 开始 Bluetooth LE 广播	129
3.4.12	AT+BLEADVSTOP: 停止 Bluetooth LE 广播	130
3.4.13	AT+BLECONN: 建立 Bluetooth LE 连接	130
3.4.14	AT+BLECONNPARAM: 查询/更新 Bluetooth LE 连接参数	131
3.4.15	AT+BLEDISCONN: 断开 Bluetooth LE 连接	132
3.4.16	AT+BLECFGMTU: 查询/设置 Bluetooth LE MTU 长度	133
3.4.17	AT+BLEGATTSSRV: GATTS 发现服务	134
3.4.18	AT+BLEGATTSSCHAR: GATTS 发现服务特征	135
3.4.19	AT+BLEGATTSENTFY: 服务器 notify 服务特征值给客户端	135
3.4.20	AT+BLEGATTSEND: 服务器 indicate 服务特征值给客户端	136
3.4.21	AT+BLEGATTSSSETATTR: GATTS 设置服务特征值	137
3.4.22	AT+BLEGATTCPRIMSRV: GATTC 发现基本服务	138
3.4.23	AT+BLEGATTCPINCLSRV: GATTC 发现包含的服务	138
3.4.24	AT+BLEGATTCCCHAR: GATTC 发现服务特征	139
3.4.25	AT+BLEGATTCCRD: GATTC 读取服务特征值	140
3.4.26	AT+BLEGATTCCWR: GATTC 写服务特征值	141
3.4.27	AT+BLESPPCFG: 查询/设置 Bluetooth LE SPP 参数	142
3.4.28	AT+BLESP: 进入 Bluetooth LE SPP 模式	143
3.4.29	AT+BLESECPARAM: 查询/设置 Bluetooth LE 加密参数	143
3.4.30	AT+BLEENC: 发起 Bluetooth LE 加密请求	145
3.4.31	AT+BLEKEYREPLY: 给对方设备回复密钥	145
3.4.32	AT+BLECONFREPLY: 给对方设备回复确认结果 (传统连接阶段)	146
3.4.33	AT+BLEENCDEV: 查询绑定的 Bluetooth LE 加密设备列表	146
3.4.34	AT+BLEENCCLEAR: 清除 Bluetooth LE 加密设备列表	147
3.4.35	AT+BLESETKEY: 设置 Bluetooth LE 静态配对密钥	148
3.4.36	AT+BLUFI: 开启或关闭 BluFi	148
3.4.37	AT+BLUFINAME: 查询/设置 BluFi 设备名称	150
3.4.38	AT+BLUFISEND: 发送 BluFi 用户自定义数据	151

3.4.39	AT+BLEPERIODICDATA: 设置 Bluetooth LE 周期性广播数据	151
3.4.40	AT+BLEPERIODICSTART: 开启周期性广播	152
3.4.41	AT+BLEPERIODICSTOP: 停止周期性广播同步	152
3.4.42	AT+BLESYNCSYNCSTART: 开启同步周期性广播	153
3.4.43	AT+BLESYNCSYNCSTOP: 停止周期性广播同步	153
3.4.44	AT+BLEREADPHY: 查询当前连接使用的 PHY	154
3.4.45	AT+BLESETPHY: 设置当前连接的 PHY	155
3.4.46	AT+BLEOTANAME: 查询/设置 Bluetooth LE OTA 设备名称	155
3.4.47	AT+BLEOTA: 初始化或反初始化 Bluetooth LE OTA	156
3.5	MQTT AT 命令集	157
3.5.1	介绍	158
3.5.2	AT+MQTTUSERCFG: 设置 MQTT 用户属性	158
3.5.3	AT+MQTTLONGCLIENTID: 设置 MQTT 客户端 ID	159
3.5.4	AT+MQTTLONGUSERNAME: 设置 MQTT 登陆用户名	159
3.5.5	AT+MQTTLONGPASSWORD: 设置 MQTT 登陆密码	160
3.5.6	AT+MQTTCONNCFG: 设置 MQTT 连接属性	161
3.5.7	AT+MQTTALPN: 设置 MQTT 应用层协议协商 (ALPN)	161
3.5.8	AT+MQTTSNI: 设置 MQTT 服务器名称指示 (SNI)	162
3.5.9	AT+MQTTCONN: 连接 MQTT Broker	163
3.5.10	AT+MQTTPUB: 发布 MQTT 消息 (字符串)	164
3.5.11	AT+MQTTPUBRAW: 发布长 MQTT 消息	164
3.5.12	AT+MQTTSUB: 订阅 MQTT Topic	165
3.5.13	AT+MQTTUNSUB: 取消订阅 MQTT Topic	166
3.5.14	AT+MQTTCLEAN: 断开 MQTT 连接	167
3.5.15	MQTT AT 错误码	167
3.5.16	MQTT AT 说明	169
3.6	HTTP AT 命令集	169
3.6.1	介绍	169
3.6.2	AT+HTTPCLIENT: 发送 HTTP 客户端请求	169
3.6.3	AT+HTTPGETSIZE: 获取 HTTP 资源大小	170
3.6.4	AT+HTTPCGET: 获取 HTTP 资源	171
3.6.5	AT+HTTPCPOST: Post 指定长度的 HTTP 数据	172
3.6.6	AT+HTTPCPUT: Put 指定长度的 HTTP 数据	172
3.6.7	AT+HTTPURLCFG: 设置/获取长的 HTTP URL	173
3.6.8	AT+HTTPCHEAD: 设置/查询 HTTP 请求头	174
3.6.9	AT+HTTPCFG: 设置 HTTP 客户端配置	175
3.6.10	HTTP AT 错误码	176
3.7	文件系统 AT 命令集	176
3.7.1	介绍	176
3.7.2	AT+FS: 文件系统操作	177
3.7.3	AT+FSMOUNT: 挂载/卸载 FS 文件系统	178
3.8	WebSocket AT 命令集	178
3.8.1	介绍	179
3.8.2	AT+WSCFG: 配置 WebSocket 参数	179
3.8.3	AT+WSHEAD: 设置/查询 WebSocket 请求头	180
3.8.4	AT+WSOPEN: 查询/打开一个 WebSocket 连接	181
3.8.5	AT+WSEND: 向 WebSocket 连接发送数据	182
3.8.6	AT+WSDATAFMT: 设置 WebSocket 接收数据格式	183
3.8.7	AT+WSCLOSE: 关闭 WebSocket 连接	183
3.9	信令测试 AT 命令	184
3.9.1	介绍	184
3.9.2	AT+FACTPLCP: 发送长 PLCP 或短 PLCP	184
3.10	驱动 AT 命令	185
3.10.1	介绍	185
3.10.2	AT+DRVADC: 读取 ADC 通道值	185
3.10.3	AT+DRVPWMINIT: 初始化 PWM 驱动器	186
3.10.4	AT+DRVPWMDUTY: 设置 PWM 占空比	187
3.10.5	AT+DRVPWMFADE: 设置 PWM 渐变	187

3.10.6	AT+DRVI2CINIT: 初始化 I2C 主机驱动	188
3.10.7	AT+DRVI2CRD: 读取 I2C 数据	188
3.10.8	AT+DRVI2CWRDATA: 写入 I2C 数据	189
3.10.9	AT+DRVI2CWRBYTES: 写入不超过 4 字节的 I2C 数据	190
3.10.10	AT+DRVSPICONFGPIO: 配置 SPI GPIO	190
3.10.11	AT+DRVSPINIT: 初始化 SPI 主机驱动	191
3.10.12	AT+DRVSPIRD: 读取 SPI 数据	192
3.10.13	AT+DRVSPIWR: 写入 SPI 数据	192
3.11	Web 服务器 AT 命令	193
3.11.1	介绍	193
3.11.2	AT+WEBSEVER: 启用或禁用通过 Web 服务器配置 Wi-Fi 连接	193
3.12	用户 AT 命令	194
3.12.1	介绍	195
3.12.2	AT+USERRAM: 操作用户的空闲 RAM	195
3.12.3	AT+USEROTA: 根据指定 URL 升级固件	196
3.12.4	AT+USERWKMCFG: 设置 AT 唤醒 MCU 的配置	197
3.12.5	AT+USERMCUSLEEP: MCU 指示自己睡眠状态	199
3.12.6	AT+USERDOCS: 查询固件对应的用户文档链接	199
3.13	AT 命令分类	200
3.14	参数信息保存在 flash 中的 AT 命令	201
3.15	AT 消息	201
3.16	AT 日志	204
4	AT 命令示例	207
4.1	AT 响应消息格式控制示例	207
4.1.1	启用系统消息过滤, 实现 HTTP 透传下载功能	207
4.2	AT+SYSMFG 命令示例	209
4.2.1	介绍	210
4.2.2	出厂参数配置	211
4.2.3	PKI 配置	218
4.2.4	GATTS 配置	225
4.2.5	自定义数据存储	231
4.3	TCP-IP AT 示例	233
4.3.1	ESP32-C6 设备作为 TCP 客户端建立单连接	234
4.3.2	ESP32-C6 设备作为 TCP 服务器建立多连接	235
4.3.3	远端 IP 地址和端口固定的 UDP 通信	237
4.3.4	远端 IP 地址和端口可变的 UDP 通信	238
4.3.5	ESP32-C6 设备作为 SSL 客户端建立单连接	240
4.3.6	ESP32-C6 设备作为 SSL 服务器建立多连接	242
4.3.7	ESP32-C6 设备作为 SSL 客户端建立双向认证单连接	244
4.3.8	ESP32-C6 设备作为 SSL 服务器建立双向认证多连接	246
4.3.9	ESP32-C6 设备作为 TCP 客户端, 建立单连接, 实现 Network 透传	248
4.3.10	ESP32-C6 设备作为 TCP 服务器, 实现 Network 透传	249
4.3.11	ESP32-C6 设备作为 softAP 在 UDP 传输中实现 Network 透传	251
4.3.12	ESP32-C6 设备获取被动接收模式下的套接字数据	253
4.3.13	ESP32-C6 设备开启 mDNS 服务, PC 通过域名连接到设备的 TCP 服务器	254
4.4	Bluetooth LE AT 示例	256
4.4.1	简介	256
4.4.2	Bluetooth LE 客户端读写服务特征值	257
4.4.3	Bluetooth LE 服务端读写服务特征值	261
4.4.4	Bluetooth LE 连接加密	265
4.4.5	两个 ESP32-C6 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	269
4.4.6	ESP32-C6 与手机建立 SPP 连接, 以及在 UART-Bluetooth LE 透传模式下传输数据	273
4.4.7	ESP32-C6 和手机之间建立 Bluetooth LE 连接并配对	275
4.5	MQTT AT 示例	276
4.5.1	基于 TCP 的 MQTT 连接 (需要本地创建 MQTT 代理) (适用于数据量少)	277
4.5.2	基于 TCP 的 MQTT 连接 (需要本地创建 MQTT 代理) (适用于数据量多)	278

4.5.3	基于 TLS 的 MQTT 连接（需要本地创建 MQTT 代理）	280
4.5.4	基于 WSS 的 MQTT 连接	281
4.6	MQTT AT 连接云示例	283
4.6.1	从 AWS IoT 获取证书以及 endpoint	283
4.6.2	使用 MQTT AT 命令基于双向认证连接 AWS IoT	284
4.7	Web Server AT 示例	286
4.7.1	使用浏览器进行 Wi-Fi 配网	286
4.7.2	使用浏览器进行 OTA 固件升级	294
4.7.3	使用微信小程序进行 Wi-Fi 配网	298
4.7.4	使用微信小程序进行 OTA 固件升级	310
4.7.5	ESP32-C6 使用 Captive Portal 功能	310
4.8	HTTP AT 示例	311
4.8.1	HTTP 客户端 HEAD 请求方法	311
4.8.2	HTTP 客户端 GET 请求方法	312
4.8.3	HTTP 客户端 POST 请求方法（适用于 POST 少量数据）	313
4.8.4	HTTP 客户端 POST 请求方法（推荐方式）	314
4.8.5	HTTP 客户端 PUT 请求方法（适用于无数据情况）	316
4.8.6	HTTP 客户端 PUT 请求方法（推荐方式）	317
4.8.7	HTTP 客户端 DELETE 请求方法	319
4.9	WebSocket AT 示例	320
4.9.1	基于 TCP 的 WebSocket 连接	320
4.9.2	基于 TLS 的 WebSocket 连接（相互鉴权）	322
4.10	Sleep AT 示例	325
4.10.1	简介	325
4.10.2	在 Wi-Fi 模式下设置为 Modem-sleep 模式	326
4.10.3	在 Wi-Fi 模式下设置为 Light-sleep 模式	327
4.10.4	在蓝牙广播态下设置为 Light-sleep 模式	328
4.10.5	在蓝牙连接态下设置为 Light-sleep 模式	328
4.10.6	设置为 Deep-sleep 模式	330
4.11	AT 配网示例	330
4.11.1	BluFi 配网	330
4.11.2	SmartConfig 配网	332
4.11.3	SoftAP 配网	333
4.11.4	WPS 配网	334
5	ESP-AT 版本简介	335
5.1	发布版本	335
5.2	我该选择哪个版本？	335
5.3	版本管理	335
5.3.1	ESP-AT 发布固件管理	335
5.3.2	ESP-AT 发布分支管理	336
5.4	支持期限	336
5.5	查看当前 AT 固件版本	338
5.6	订阅 AT 版本发布	338
5.6.1	第一步：登录你的 GitHub 账号	338
5.6.2	第二步：选择定制化的通知	338
5.6.3	第三步：定制发布应用	338
6	如何编译和开发自己的 AT 工程	341
6.1	本地编译 ESP-AT 工程	341
6.1.1	详细步骤	341
6.1.2	第一步：ESP-IDF 快速入门	341
6.1.3	第二步：获取 ESP-AT	342
6.1.4	第三步：安装环境	342
6.1.5	第四步：连接设备	343
6.1.6	第五步：配置工程	343
6.1.7	第六步：编译工程	343
6.1.8	第七步：烧录到设备	344

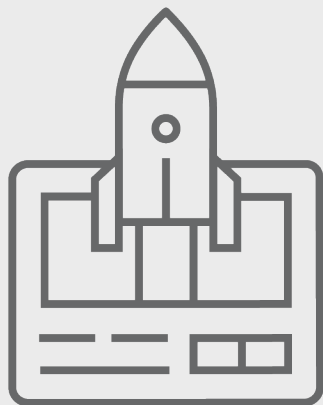
6.1.9	build.py 进阶用法	344
6.2	网页编译 ESP-AT 工程	345
6.2.1	详细步骤	345
6.2.2	第一步：登录你的 GitHub 账号	345
6.2.3	第二步：Fork ESP-AT 工程	345
6.2.4	第三步：开启 GitHub Actions 功能	345
6.2.5	第四步：配置编译 ESP-AT 工程所需的密钥	350
6.2.6	第五步：使用 github.dev 编辑器修改和提交代码	354
6.2.7	第六步：GitHub Actions 编译 AT 固件	359
6.3	如何设置 AT 端口管脚	359
6.3.1	ESP32-C6 系列	359
6.4	添加自定义 AT 命令	360
6.4.1	自定义 AT 命令	360
6.4.2	运行 AT+TEST 命令获取运行结果	363
6.4.3	自定义复杂的 AT 命令	364
6.5	如何提高 ESP-AT 吞吐性能	372
6.5.1	(简单) 快速配置	372
6.5.2	(推荐) 熟悉数据流、针对性地配置	373
6.6	如何更新 mfg_nvs 分区	375
6.6.1	mfg_nvs 分区介绍	375
6.6.2	生成 mfg_nvs.bin	375
6.6.3	下载 mfg_nvs.bin	375
6.7	如何更新出厂参数	376
6.7.1	出厂参数配置介绍	376
6.7.2	生成 mfg_nvs.bin 文件	377
6.7.3	下载 mfg_nvs.bin 文件	377
6.8	如何更新 PKI 配置	377
6.8.1	PKI 配置介绍	378
6.8.2	生成 mfg_nvs.bin 文件	378
6.8.3	下载 mfg_nvs.bin 文件	379
6.9	如何自定义低功耗蓝牙服务	379
6.9.1	低功耗蓝牙服务源文件介绍	379
6.9.2	编译时自定义低功耗蓝牙服务	381
6.10	如何自定义分区	383
6.10.1	修改 at_customize.csv	383
6.10.2	生成 at_customize.bin	384
6.10.3	烧录 at_customize.bin 至 ESP32-C6 设备	384
6.10.4	示例	384
6.11	如何增加一个新的模组支持	385
6.11.1	第一步：配置新增模组的出厂参数	385
6.11.2	第二步：配置新增模组的 OTA	386
6.11.3	第三步：新增模组配置	386
6.12	ESP32-C6 SDIO AT 指南	388
6.12.1	简介	388
6.12.2	如何使用 SDIO AT	389
6.12.3	SDIO 交互流程	389
6.13	SPI AT 指南	390
6.13.1	简介	390
6.13.2	使用 SPI AT	391
6.13.3	SPI AT 速率	393
6.14	如何实现 OTA 升级	394
6.14.1	OTA 命令对比及应用场景	394
6.14.2	使用 ESP-AT OTA 命令执行 OTA 升级	396
6.15	如何实现 Bluetooth LE OTA 升级	401
6.15.1	Bluetooth LE OTA 用法	402
6.15.2	Bluetooth LE OTA 原理	403
6.15.3	参考	405
6.16	如何更新 ESP-IDF 版本	406

6.17	ESP-AT 固件差异	406
6.18	如何从 GitHub 下载最新临时版本 AT 固件	406
6.19	at.py 工具	411
6.19.1	详细步骤	411
6.19.2	第一步: Python 安装	411
6.19.3	第二步: at.py 下载	411
6.19.4	第三步: at.py 用法说明	411
6.19.5	第四步: at.py 修改固件中的配置示例	411
6.20	AT API Reference	416
6.20.1	Header File	416
6.20.2	Functions	416
6.20.3	Structures	420
6.20.4	Macros	422
6.20.5	Type Definitions	423
6.20.6	Enumerations	423
6.20.7	Header File	426
6.20.8	Functions	426
6.20.9	Macros	428
6.21	如何配置静默模式	429
6.21.1	简介	429
6.21.2	配置	430
6.22	如何启用更多 AT 调试日志	431
6.22.1	AT Wi-Fi 功能调试	431
6.22.2	AT 网络功能调试	431
6.22.3	AT BLE 功能调试	433
6.22.4	AT 接口功能调试	433
6.22.5	调试示例	433
7	第三方开放云平台定制化 AT 命令和固件	435
8	AT FAQ	437
8.1	AT 文档	438
8.1.1	如何选择正确的 AT 文档版本?	438
8.2	AT 固件	438
8.2.1	我的模组没有官方发布的固件, 如何获取适用的固件?	438
8.2.2	如何获取 AT 固件源码?	438
8.2.3	官网上放置的 AT 固件如何下载?	439
8.2.4	如何整合 ESP-AT 编译出来的所有 bin 文件?	439
8.2.5	新购买的 ESP 系列模组上电后, 串口打印错误 “flash read err” 是什么原因? 该模组如何使用 AT 命令?	439
8.2.6	模组出厂 AT 固件是否支持流控?	439
8.3	AT 命令与响应	439
8.3.1	AT 提示 busy p... 是什么原因?	439
8.3.2	AT 固件, 上电后发送第一个命令总是会返回下面的信息, 为什么?	439
8.3.3	在不同模组上的默认 AT 固件支持哪些命令, 以及哪些命令从哪个版本开始支持?	439
8.3.4	MCU 给 ESP32-C6 设备发 AT 命令无返回, 是什么原因?	440
8.3.5	AT 命令中特殊字符如何处理?	440
8.3.6	AT 命令中串口波特率是否可以修改? (默认: 115200)	440
8.3.7	如何修改 AT 命令端口的 UART 管脚?	440
8.3.8	如何解决 TCP/SSL 数据传输时的丢失问题?	440
8.3.9	ESP32-C6 使用 AT 命令进入透传模式, 如果连接的热点断开, ESP32-C6 能否给出相应的提示信息?	440
8.3.10	Wi-Fi 为什么断开 (WIFI DISCONNECT)?	440
8.3.11	Wi-Fi 常见的兼容性问题有哪些?	441
8.3.12	AT 命令是否支持 ESP-WIFI-MESH?	441
8.3.13	是否有 AT 命令连接阿里云以及腾讯云示例?	441
8.3.14	AT 命令是否可以设置低功耗蓝牙发射功率?	441
8.3.15	低功耗蓝牙客户端如何使能 notify 和 indicate 功能?	441

8.4	硬件	441
8.4.1	在不同模组上的 AT 固件要求芯片 flash 多大?	441
8.4.2	AT 固件如何查看 error log?	442
8.4.3	AT 在 ESP32-C6 模组上的 UART1 通信管脚与 ESP32-C6 模组的 datasheet 默认 UART1 管脚不一致?	442
8.5	性能	442
8.5.1	AT Wi-Fi 连接耗时多少?	442
8.5.2	AT 固件中 TCP 发送窗口大小是否可以修改?	442
8.5.3	AT 吞吐量如何测试及优化?	442
8.5.4	如何修改 AT 默认 TCP 数据段最大重传次数?	442
8.6	其他	442
8.6.1	乐鑫芯片可以通过哪些接口来传输 AT 命令?	442
8.6.2	AT 如何使能调试日志?	442
8.6.3	如何修改 AT 固件首次启动时的默认 Wi-Fi 模式?	443
8.6.4	AT 指令如何实现 HTTP 断点续传功能?	443
8.6.5	如何从 HTTP 服务器下载文件并存储到 FATFS, 或将 FATFS 文件系统中的文件上传到 HTTP 服务器?	443
8.6.6	使用 nmap 等工具扫描发现 AT 设备存在开放端口, 是否说明固件存在安全风险?	443
8.6.7	使用 esp-idf-sbom 扫描 AT 固件出现大量 CVE, 是否会影响 RED、EN 18031、CRA 等合规?	443
9	Index of Abbreviations	445
10	特别声明	451
10.1	应用场景验证	451
10.2	兼容性	451
10.3	服务器升级注意事项	451
10.4	无线方案问题处理	451
10.5	使用限制	451
10.6	责任限制	452
10.7	第三方软件和技术	452
10.8	数据保护	452
10.9	版本更新与维护	452
10.10	声明更新	452
10.11	用户责任	452
11	关于 ESP-AT	453
11.1	其他 Espressif 连接方案	453
	索引	455
	索引	455

这里是乐鑫 ESP-AT 开发框架的文档中心。ESP-AT 作为由 Espressif Systems 发起和提供技术支持的官方项目，适用于 Windows、Linux、macOS 上的 ESP32、ESP32-C2、ESP32-C3、ESP32-C5、ESP32-C6、ESP32-C61、和 ESP32-S2 系列芯片。

本文档仅包含针对 ESP32-C6 芯片的 ESP-AT 使用。



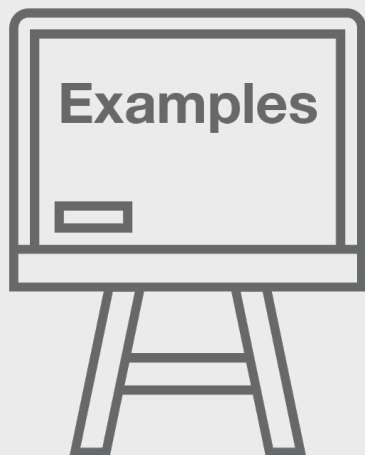
[入门](#)



[AT Binary 列表](#)



[AT 命令集](#)



[AT 命令示例](#)



[编译和开发](#)



[第三方定制化 AT 命令和固件](#)

Chapter 1

入门指南

本指南详细介绍 ESP-AT 是什么、技术选型、如何连接硬件、以及如何下载和烧录 AT 固件，由以下章节组成：

1.1 ESP-AT 是什么

重要：在使用 ESP-AT 前，请仔细阅读[特别声明](#)，并遵循其中的各项条款和注意事项。

ESP-AT 是乐鑫开发的可直接用于量产的物联网应用固件，旨在降低客户开发成本，快速形成产品。通过 ESP-AT 命令，你可以快速加入无线网络、连接云平台、实现数据通信以及远程控制等功能，真正的通过无线通讯实现万物互联。

ESP-AT 是基于 ESP-IDF 实现的软件工程。它使 ESP32-C6 模组作为从机，MCU 作为主机。MCU 发送 AT 命令给 ESP32-C6 模组，控制 ESP32-C6 模组执行不同的操作，并接收 ESP32-C6 模组返回的 AT 响应。ESP-AT 提供了大量功能不同的 AT 命令，如 Wi-Fi 命令、TCP/IP 命令、Bluetooth LE 命令、MQTT 命令、HTTP 命令等。

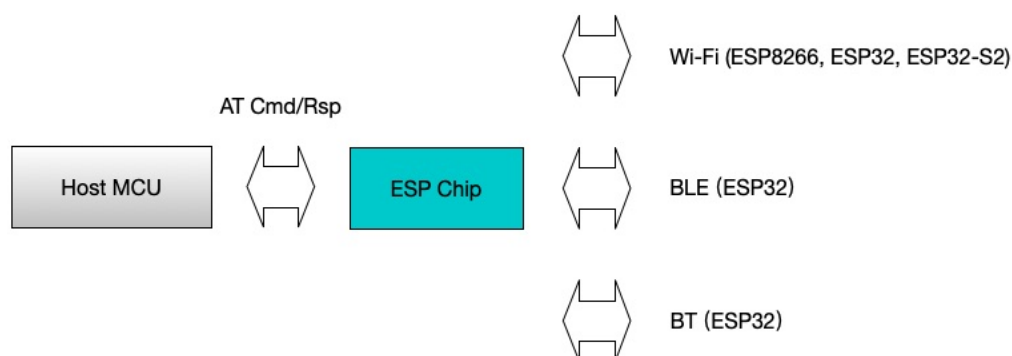


图 1: ESP-AT 概览

AT 命令以 “AT” 开始，代表 Attention，以新的一行 (CR LF) 为结尾。输入的每条命令都会返回 OK 或 ERROR 的响应，表示当前命令的最终执行结果。注意，所有 AT 命令均为串行执行，每次只能执行一条

命令。因此，在使用 AT 命令时，应等待上一条命令执行完毕后，再发送下一条命令。如果上一条命令未执行完毕，又发送了新的命令，则会返回 `busy p...` 提示。更多有关 AT 命令的信息可参见 [AT 命令集](#)。

默认配置下，MCU 通过 UART 连接至 ESP32-C6 模组、发送 AT 命令以及接收 AT 响应。但是，你也可以根据实际使用情况修改程序，使用其他的通信接口，例如 SDIO。

同样，你也可以基于 ESP-AT 工程，自行开发更多的 AT 命令，以实现更多的功能。

1.2 技术选型

本文档主要介绍如何选择合适的乐鑫硬件产品、AT 软件方案、以及项目初期的准备。

重要： 如果你在选择乐鑫硬件产品、AT 软件方案中有任何问题，请联系 [乐鑫商务](#) 或者 [技术支持](#)。

1.2.1 硬件选型

在开始使用 ESP-AT 之前，你需要选择一款合适的乐鑫芯片集成到你的产品中，为你的产品赋能无线功能。硬件选型是一个复杂的过程，需要考虑多方面的因素，如功能、功耗、成本、尺寸等。请阅读下面内容帮助你选型硬件。

- [产品选型工具](#) 可以帮助你了解不同乐鑫产品的硬件区别。
- [技术规格书](#) 可以帮助你了解该芯片/模组所支持的硬件能力。
- [硬件选型入门指导](#) 可以帮助你简要对比芯片差别，了解芯片、模组、和开发板的差别以及选择指南。

备注： 技术规格书中载明的是硬件最大能力，不代表 AT 软件能力。例如，ESP32-C6 芯片支持 Zigbee 3.0 及 Thread 1.3，但是现有的 AT 软件方案暂未支持这两种无线协议。

1.2.2 AT 软件方案选型

AT 软件方案是乐鑫针对不同芯片提供的 AT 固件，可以帮助你快速实现无线功能。

- 如果你想了解 ESP32-C6 芯片详细的 AT 软件能力，请参考 [AT 命令集](#)。
- 如果你想对比 ESP32-C6 芯片不同的 AT 固件功能，请参考 [ESP-AT 固件差异](#)。

下表为 AT 固件当前支持和计划支持的乐鑫芯片列表及其简要对比。无计划支持的芯片不在此表中展示。

芯片	无线功能	推荐的 AT 固件	说明
ESP32-C6	Wi-Fi 6 + BLE 5.0	v4.2.0.0	
ESP32-C5	5G Wi-Fi + Wi-Fi 6 + BLE 5.0	v5.0.1.0	
ESP32-C61	Wi-Fi 6 + BLE 5.0	v5.0.1.0	
ESP32-C3	Wi-Fi 4 + BLE 5.0	v4.2.0.0	
ESP32-C2	Wi-Fi 4 (或 BLE 5.0)	v4.2.0.0	
ESP32	Wi-Fi 4 + BLE v4.2 (+ BT)	v4.2.0.0	
ESP32-S2	Wi-Fi 4	v4.2.0.0	推荐使用性价比更高的 ESP32-C 系列

- (或 BLE 5.0) 表示 AT 软件方案中支持低功耗蓝牙功能，但发布的固件中未包含此功能。
- (+ BT) 表示 AT 软件方案中支持经典蓝牙功能，但发布的固件中未包含此功能。

备注： 出厂的模组或芯片中，均未烧录 AT 固件。若你有量产需求，请及时联系对接的商务人员或 sales@espressif.com，我们将提供定制生产。

我该选哪种类型的 AT 固件？

ESP-AT 固件有以下几种类型，其中下载或准备固件的工作量自上而下依次递增，支持的模组类型也自上而下依次递增。

- 官方发布版固件（推荐）
- [GitHub](#) 临时固件
- 修改参数的固件
- 自行编译的固件

官方发布版固件（推荐） 官方发布版固件又称“发布版固件”、“官方固件”、“默认固件”，为乐鑫官方团队测试通过并发布的固件，固件会根据内部开发计划周期性发布，此种固件可直接基于乐鑫 OTA 服务器升级固件。如果 官方发布版固件完全满足你的项目需求，建议你优先选择 官方发布版固件。如果官方固件不支持你的模组，你可以根据 [硬件差异](#)，选择和你的模组硬件配置相近的固件进行测试验证。

- 获取途径：[ESP32-C6 AT 固件](#)
- 优点
 - 稳定
 - 可靠
 - 获取固件工作量小
- 缺点
 - 更新周期长
 - 覆盖的模组有限
- 参考文档
 - [硬件连接](#)
 - [固件下载及烧录指南](#)
 - 有关 ESP-AT 固件支持/不支持哪些芯片系列，请参考 ESP-AT GitHub 首页 [readme.md](#)

GitHub 临时固件 GitHub 临时固件为每次将代码推送到 GitHub 时都会生成但并未达到固件发布周期条件的固件，或者说是开发中的固件，包括 官方发布版固件的临时版本和适配过但是不计划正式发布的固件，其中前者可直接基于乐鑫 OTA 服务器升级固件。

- 获取途径：请参考[如何从 GitHub 下载最新临时版本 AT 固件](#)。
- 优点：
 - 实时性强，新的特性和漏洞修补都会实时同步出来。
 - 包含一些非正式发布的固件，如基于 SDIO 通讯的固件、基于 SPI 通讯的固件。
 - 获取固件工作量小。
- 缺点：基于非正式发布的 commit 生成的固件未经过完整的测试，可能会存在一些风险，需要你自己做完整的测试。

修改参数的固件 修改参数的固件指的是只修改参数区域而并不需要重新编译的固件，适用于固件功能满足项目要求、但只有某些参数不满足的情况下，如 UART 波特率、UART GPIO 管脚等参数的变更，此种固件可直接基于乐鑫 OTA 服务器升级固件。

- 关于如何修改参数文件，请参考[at.py 工具](#)。
- 优点：
 - 不需要重新编译固件。
 - 固件稳定、可靠。
- 缺点：需要基于发布版的固件修改，更新周期长，覆盖的模组有限。

自行编译的固件 当你需要进行二次开发时可采用此种方式。需要自己部署 OTA 服务器以支持 OTA 功能。

- 关于如何自行编译固件，请参考[本地编译 ESP-AT 工程](#)。
- 优点：功能、周期自己可控。
- 缺点：需要自己搭建环境编译。

如果你希望稳定性优先，推荐基于该芯片最新已发布的版本对应的分支开发你的 AT 固件。如果你希望更多新功能，推荐基于 [master 分支](#) 开发你的 AT 固件。

1.2.3 项目初期准备

项目初期准备阶段，**强烈建议**你选择 [乐鑫开发板](#) 开始你的项目。在项目初期，能够帮助你快速原型验证，评估硬件和软件能力，减少项目风险；在项目中期，能够帮助你快速功能集成和验证，性能优化，提高开发效率；在项目后期，能够帮助你快速模拟和定位问题，实现产品快速迭代。

如果你是[自行编译的固件](#)，建议你优先选择 Linux 系统作为开发环境。

1.3 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。

不同系列的 AT 固件支持的命令不同，适用的模组或芯片也不尽相同，详情可参考[ESP-AT 固件差异](#)。

如果不想使用 AT 默认管脚，可以参考[如何设置 AT 端口管脚](#) 文档更改管脚。

1.3.1 硬件准备

表 1: ESP-AT 测试所需硬件

硬件	功能
ESP32-C6 开发板	从机
USB 数据线（连接 ESP32-C6 开发板和 PC）	下载固件、输出日志数据连接
PC	主机，将固件下载至从机
USB 数据线（连接 PC 和 USB 转 UART 串口模块）	发送 AT 命令、接收 AT 响应数据连接
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 ESP32-C6 开发板）	发送 AT 命令、接收 AT 响应数据连接

注意：

- 上图使用 4 根杜邦线连接 ESP32-C6 开发板和 USB 转 UART 串口模块，但如果你不使用硬件流控功能，只需 2 根杜邦线连接 TX/RX 即可。
- 如果你使用 ESP32-C6 模组（而非开发板）并通过 UART 烧录固件，需要：
 - 预留 U0RXD 和 U0TXD 管脚（参考 [技术规格书 > IO 管脚](#)）
 - 预留 Strapping 管脚（参考 [技术规格书 > 启动配置项](#)），通过控制 Strapping 管脚电平进入下载模式
 - 更多细节请参考 [ESP32-C6 进入下载模式的硬件要求](#)

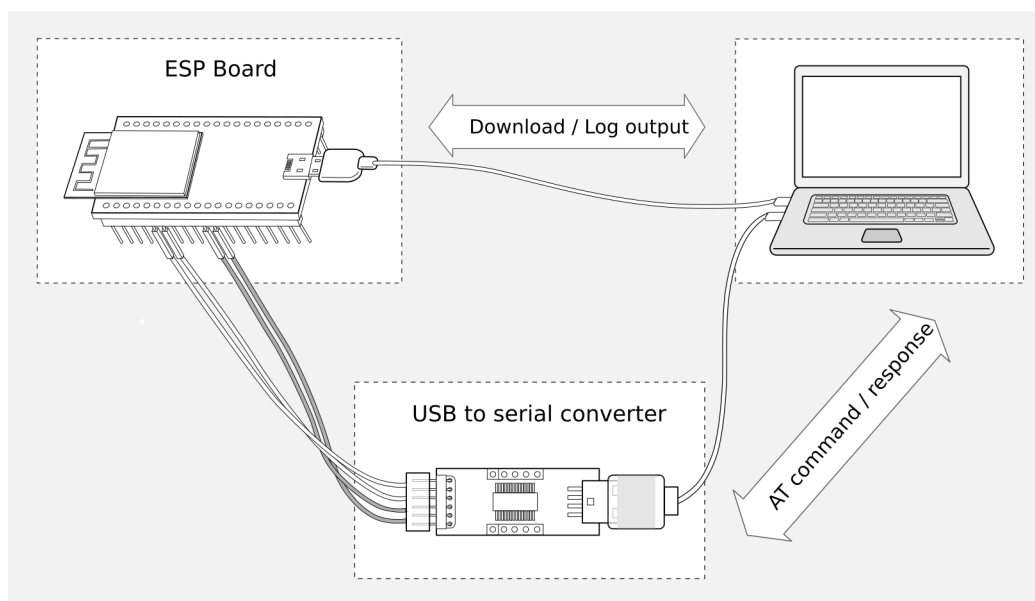


图 2: ESP-AT 测试硬件连接示意图

1.3.2 ESP32-C6-4MB 系列

ESP32-C6-4MB 系列指的是内置 ESP32-C6 芯片，同时有 4 MB flash 的模组/开发板，例如：ESP32-C6 MINI 系列设备、ESP32-C6 WROOM 系列设备。

ESP32-C6-4MB 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。

表 2: ESP32-C6-4MB 系列硬件连接管脚分配

功能	ESP32-C6-4MB 开发板/模组管脚	其它设备管脚
下载固件/输出日志 ¹	UART0 • GPIO17 (RX) • GPIO16 (TX)	PC • TX • RX
AT 命令/响应 ²	UART1 • GPIO6 (RX) • GPIO7 (TX) • GPIO5 (CTS) • GPIO4 (RTS)	USB 转 UART 串口模块 • TX • RX • RTS • CTS

说明 1：ESP32-C6-4MB 开发板和 PC 之间的管脚连接已内置在 ESP32-C6-4MB 开发板上，你只需使用 USB 数据线连接开发板和 PC 即可。

说明 2：CTS/RTS 管脚只有在使用硬件流控功能时才需连接。

如果需要直接基于 ESP32-C6-4MB 模组进行连接，请参考对应模组的 [技术规格书](#)。

1.4 下载指导

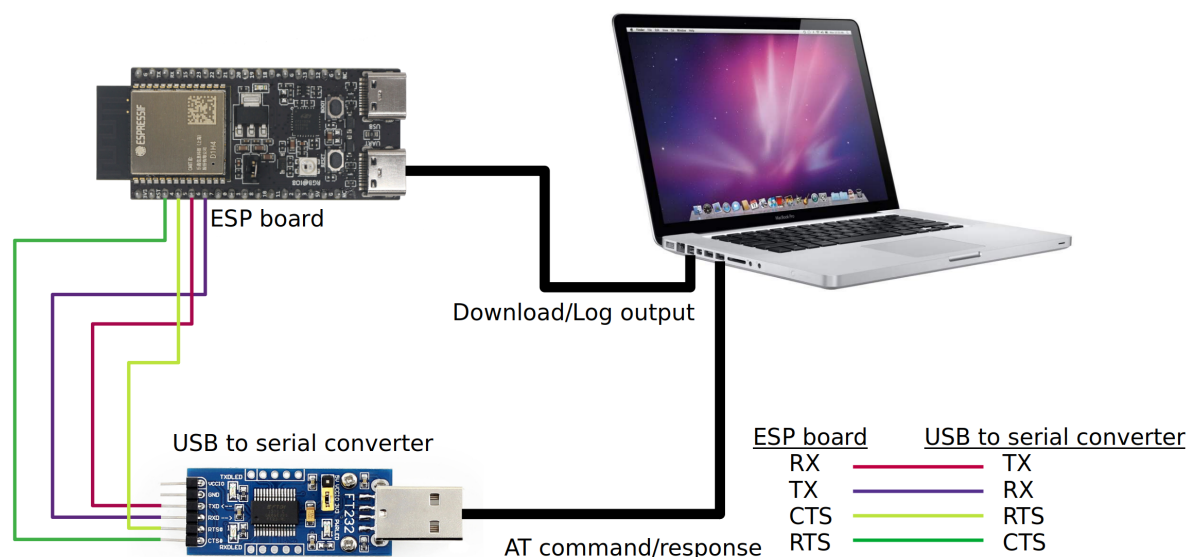


图 3: ESP32-C6-4MB 系列硬件连接示意图

本文档以 ESP32-C6-MINI-1 模组为例，介绍如何下载 ESP32-C6-MINI-1 模组对应的 AT 固件，以及如何将固件烧录到模组上，其它 ESP32-C6 系列模组也可参考本文档。

下载和烧录 AT 固件之前，请确保你已正确连接所需硬件，具体可参考[硬件连接](#)。

对于不同系列的模组，AT 默认固件所支持的命令会有所差异。具体可参考[ESP-AT 固件差异](#)。

1.4.1 下载 AT 固件

请按照以下步骤将 AT 固件下载至 PC：

- 前往[AT 固件](#)
- 找到你的模组所对应的 AT 固件
- 点击相应链接进行下载

此处，我们下载了 ESP32-C6-MINI-1 对应的 ESP32C6-4MB-AT-V4.0.0.0 固件，该固件的目录结构及其中各个 bin 文件介绍如下，其它 ESP32-C6 系列模组固件的目录结构及 bin 文件也可参考如下介绍：

```
.
├── at_customize.bin           // 二级分区表
├── bootloader                 // bootloader
│   └── bootloader.bin
├── customized_partitions      // AT 自定义 bin 文件
│   └── mfg_nvs.csv           // 量产 NVS 分区的原始数据
│   └── mfg_nvs.bin           // 量产 NVS 分区 bin 文件
├── download.config            // 烧录固件的参数
├── esp-at.bin                 // AT 应用固件
├── esp-at.elf
├── esp-at.map
├── factory                    // 量产所需打包好的 bin 文件
│   └── factory_XXX.bin
├── flasher_args.json          // 下载参数信息新的格式
├── ota_data_initial.bin       // ota data 区初始值
├── partition_table            // 一级分区列表
│   └── partition-table.bin
└── sdkconfig                  // AT 固件对应的编译配置
```

其中，download.config 文件包含烧录固件的参数：

```
--flash_mode dio --flash_freq 80m --flash_size 4MB
0x0 bootloader/bootloader.bin
0x8000 partition_table/partition-table.bin
0xd000 ota_data_initial.bin
0x1e000 at_customize.bin
0x1f000 customized_partitions/mfg_nvs.bin
0x60000 esp-at.bin
```

- `--flash_mode dio` 代表此固件采用的 flash dio 模式进行编译；
- `--flash_freq 80m` 代表此固件采用的 flash 通讯频率为 80 MHz；
- `--flash_size 4MB` 代表此固件适用的 flash 最小为 4 MB；
- `0xd000 ota_data_initial.bin` 代表在 `0xd000` 地址烧录 `ota_data_initial.bin` 文件。

1.4.2 烧录 AT 固件至设备

请根据你的操作系统选择对应的烧录方法。

Windows

开始烧录之前，请下载 Windows [Flash 下载工具](#)，详细指导可参阅 [Flash 下载工具用户指南](#)。

- 打开 Flash 下载工具
- 选择芯片类型（此处，我们选择 ESP32-C6）
- 根据你的需求选择一种工作模式（此处，我们选择 develop）
- 根据你的需求选择一种下载接口（此处，我们选择 uart）

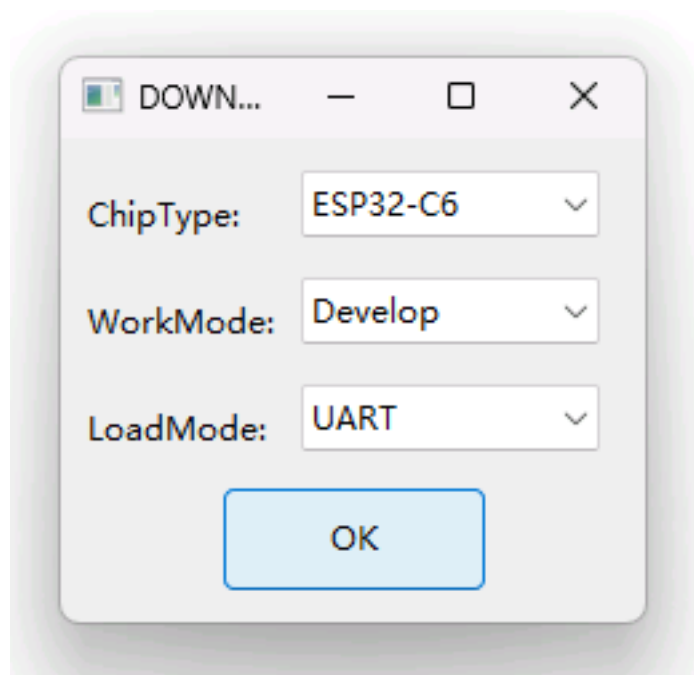


图 4: 固件下载配置选择

- 将 AT 固件烧录至设备，以下两种方式任选其一：
 - 直接下载打包好的量产固件（即 `build/factory` 目录下的 `factory_XXX.bin`）至 `0x0` 地址：勾选 “DoNotChgBin”，使用量产固件的默认配置；
 - 分开下载多个 `bin` 文件至不同的地址：根据 `download.config` 文件进行配置，请勿勾选 “DoNotChgBin”；

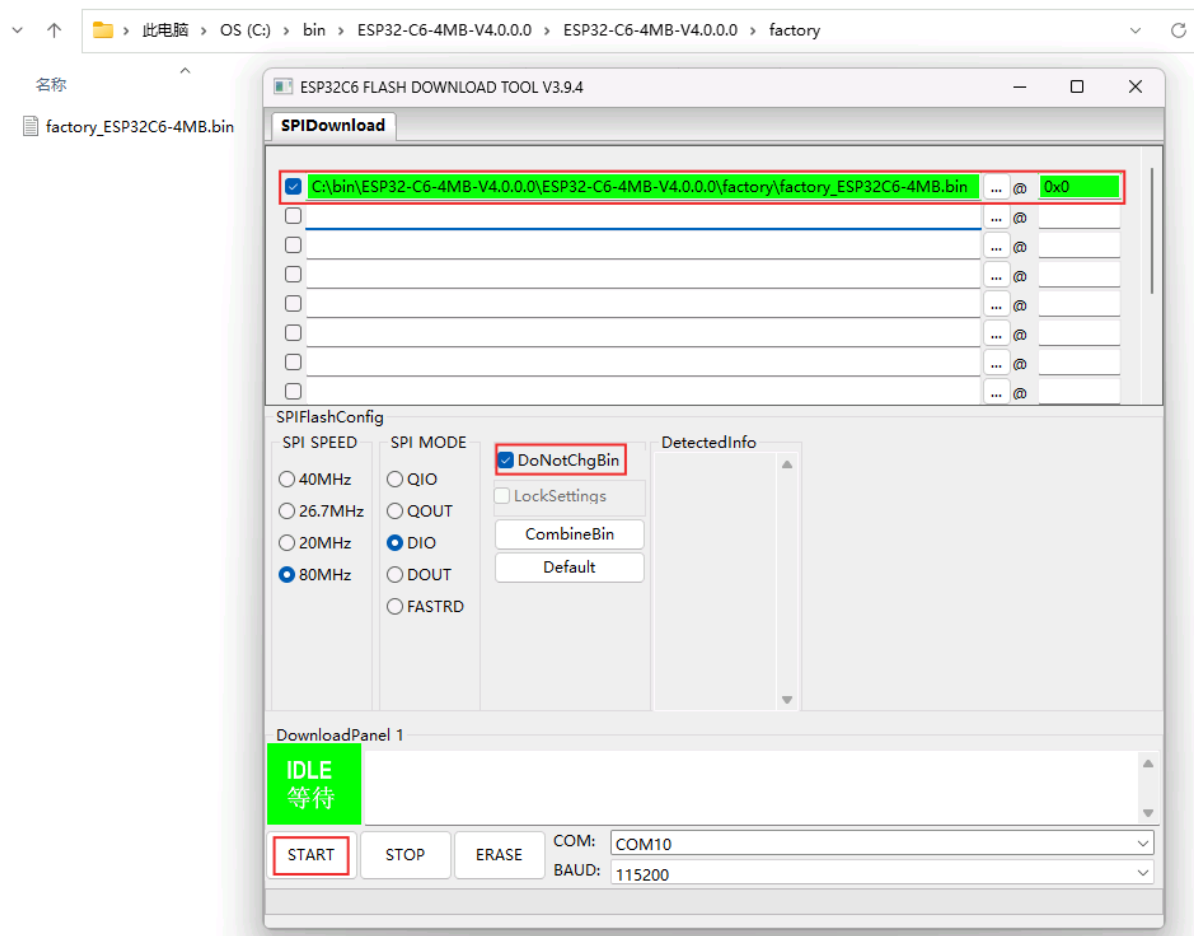


图 5: 下载至单个地址界面图 (点击放大)

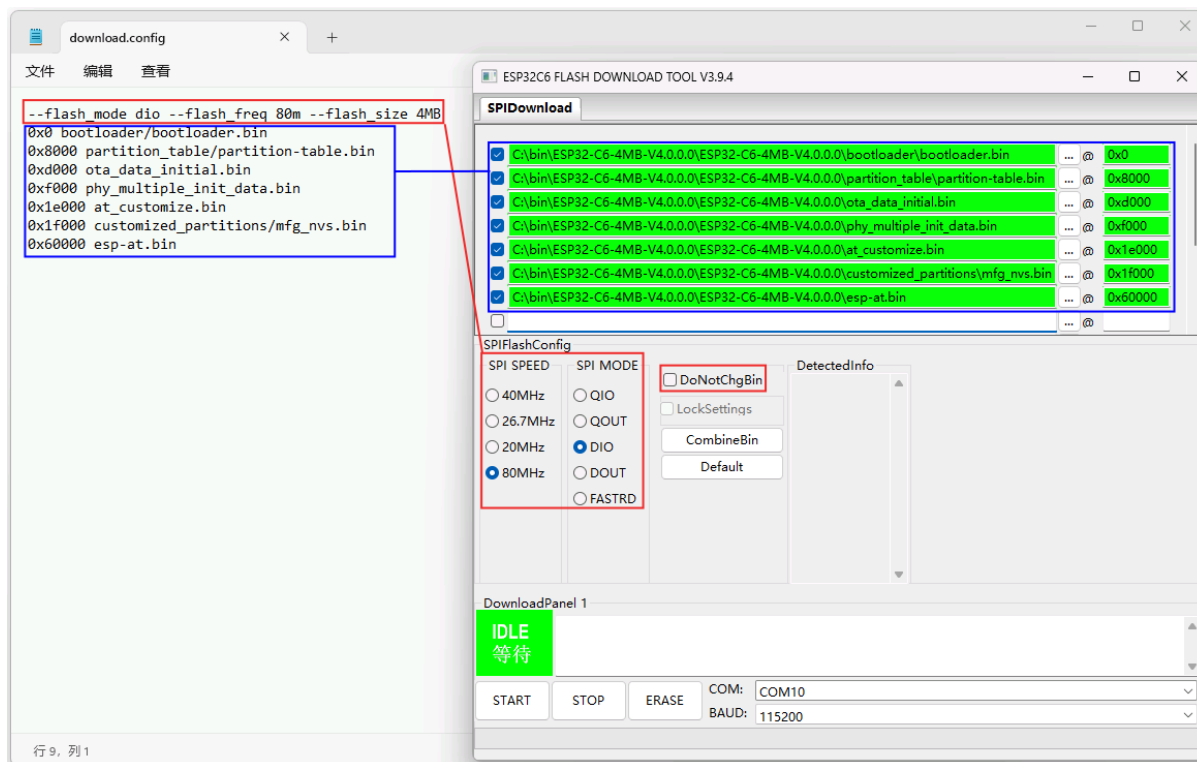


图 6: 下载至多个地址界面图 (点击放大)

为了避免烧录出现问题，请查看开发板的下载接口的 COM 端口号，并从“COM:”下拉列表中选择该端口号。有关如何查看端口号的详细介绍请参考 [在 Windows 上查看端口](#)。

烧录完成后，请[检查 AT 固件是否烧录成功](#)。

Linux 或 macOS

开始烧录之前，请安装 `esptool.py`。

以下两种方式任选其一，将 AT 固件烧录至设备：

- 分开下载多个 bin 文件至不同的地址：输入以下命令，替换 PORTNAME 和 download.config 参数；

```
esptool.py --chip auto --port PORTNAME --baud 115200 --before default_reset --
↳after hard_reset write_flash -z download.config
```

将 PORTNAME 替换成开发板的下载接口名称，若你无法确定该接口名称，请参考 [在 Linux 和 macOS 上查看端口](#)。

将 download.config 替换成该文件里的参数列表。

以下是将固件烧录至 ESP32-C6-MINI-1 模组输入的命令：

```
esptool.py --chip auto --port /dev/tty.usbserial-0001 --baud 115200 --
↳before default_reset --after hard_reset write_flash -z --flash_mode_
↳dio --flash_freq 80m --flash_size 4MB 0x8000 partition_table/
↳partition-table.bin 0xd000 ota_data_initial.bin 0x0 bootloader/
↳bootloader.bin 0x60000 esp-at.bin 0x1e000 at_customize.bin 0x1f000_
↳customized_partitions/mfg_nvs.bin
```

- 直接下载打包好的量产固件至 0x0 地址：输入以下命令，替换 PORTNAME 和 FILEDIRECTORY 参数；

```
esptool.py --chip auto --port PORTNAME --baud 115200 --before default_reset --
↳after hard_reset write_flash -z --flash_mode dio --flash_freq 40m --flash_
↳size 4MB 0x0 FILEDIRECTORY
```

将 PORTNAME 替换成开发板的下载接口名称，若你无法确定该接口名称，请参考 [在 Linux 和 macOS 上查看端口](#)。

将 FILEDIRECTORY 替换成打包好的量产固件的文件路径，通常情况下，文件路径是 factory/XXX.bin。

以下是将固件烧录至 ESP32-C6-MINI-1 模组输入的命令：

```
esptool.py --chip auto --port /dev/tty.usbserial-0001 --baud 115200 --
↳before default_reset --after hard_reset write_flash -z --flash_mode_
↳dio --flash_freq 80m --flash_size 4MB 0x0 factory/factory_ESP32C6-
↳4MB.bin
```

烧录完成后，请[检查 AT 固件是否烧录成功](#)。

1.4.3 检查 AT 固件是否烧录成功

请按照以下步骤检查 AT 固件是否烧录成功：

- 打开串口工具，如 SecureCRT；
- 串口：选择用于发送或接收“AT 命令/响应”的串口（详情请见[硬件连接](#)）；
- 波特率：115200；
- 数据位：8；
- 检验位：None；
- 停止位：1；
- 流控：None；
- 输入“AT+GMR”命令，并且换行 (CR LF)；

若如下所示，响应是 OK，则表示 AT 固件烧录成功。

```

AT+GMR
AT version:4.0.0.0(3fe3806 - ESP32C6 - Dec 29 2023 11:10:21)
SDK version:v5.1.2-dirty
compile time(89040be7):Jan 2 2024 05:53:07
Bin version:v4.0.0.0(ESP32C6-4MB)

OK

```

否则，你需要通过以下方式之一检查 ESP32-C6 设备开机日志：

方法 1:

- 打开串口工具，如 SecureCRT；
- 串口：选择用于“下载固件/输出日志”的串口，串口详情请参阅[硬件连接](#)。
- 波特率：115200；
- 数据位：8；
- 检验位：None；
- 停止位：1；
- 流控：None；
- 直接按开发板的 RST 键，若日志和下面的日志相似，则说明 ESP-AT 固件已经正确初始化了。

方法 2:

- 打开两个串口工具，如 SecureCRT；
- 串口：分别选择用于发送或接收“AT 命令/响应”的串口以及用于“下载固件/输出日志”的串口，串口详情请参阅[硬件连接](#)。
- 波特率：115200；
- 数据位：8；
- 检验位：None；
- 停止位：1；
- 流控：None；
- 在发送或接收“AT 命令/响应”的串口输入 **AT+RST** 命令，并且换行 (CR LF)，若“下载固件/输出日志”的串口日志和下面的日志相似，则说明 ESP-AT 固件已经正确初始化了。

ESP32-C6 开机日志：

```

ESP-ROM:esp32c6-20220919
Build:Sep 19 2022
rst:0xc (SW_CPU),boot:0x6c (SPI_FAST_FLASH_BOOT)
Saved PC:0x4001975a
SPIWP:0xee
mode:DIO, clock div:2
load:0x4086c410,len:0xd50
load:0x4086e610,len:0x2d74
load:0x40875720,len:0x1800
entry 0x4086c410
I (27) boot: ESP-IDF v5.0-dev-9643-g4bc762621d-dirty 2nd stage bootloader
I (27) boot: compile time Jul 5 2023 11:12:16
I (29) boot: chip revision: v0.1
I (32) boot.esp32c6: SPI Speed      : 40MHz
I (37) boot.esp32c6: SPI Mode      : DIO
I (41) boot.esp32c6: SPI Flash Size : 4MB
I (46) boot: Enabling RNG early entropy source...
I (52) boot: Partition Table:
I (55) boot: ## Label                Usage            Type ST Offset   Length
I (62) boot:  0 otadata              OTA data         01 00 0000d000 00002000
I (70) boot:  1 phy_init              RF data          01 01 0000f000 00001000
I (77) boot:  2 nvs                   WiFi data        01 02 00010000 0000e000
I (85) boot:  3 at_customize          unknown          40 00 0001e000 00042000
I (92) boot:  4 ota_0                 OTA app          00 10 00060000 001d0000
I (100) boot:  5 ota_1                 OTA app          00 11 00230000 001d0000
I (107) boot: End of partition table

```

(续下页)

(接上页)

```
I (112) esp_image: segment 0: paddr=00060020 vaddr=42140020 size=30628h (198184) ↵  
↵map  
I (198) esp_image: segment 1: paddr=00090650 vaddr=40800000 size=0f9c8h ( 63944) ↵  
↵load  
I (228) esp_image: segment 2: paddr=000a0020 vaddr=42000020 size=13c688h (1296008) ↵  
↵map  
I (740) esp_image: segment 3: paddr=001dc6b0 vaddr=4080f9c8 size=05bf4h ( 23540) ↵  
↵load  
I (752) esp_image: segment 4: paddr=001e22ac vaddr=408155c0 size=03c54h ( 15444) ↵  
↵load  
I (760) esp_image: segment 5: paddr=001e5f08 vaddr=50000000 size=00068h (   104) ↵  
↵load  
I (771) boot: Loaded app from partition at offset 0x60000  
I (772) boot: Disabling RNG early entropy source...  
no external 32k oscillator, disable it now.  
at param mode: 1  
AT cmd port:uart1 tx:7 rx:6 cts:5 rts:4 baudrate:115200  
module_name: ESP32C6-4MB  
max tx power=78, ret=0  
4.0.0
```

如果你尚不了解 ESP-AT 工程，请阅读[ESP-AT 是什么](#)。

如果你想学习如何使用 ESP-AT，请首先阅读[技术选型](#)，帮助你选型硬件和软件，再阅读[硬件连接](#)，了解所需的硬件以及硬件之间如何连接，然后再阅读[下载指导](#)，了解如何下载和烧录 AT 固件。

Chapter 2

AT 固件

2.1 发布的固件

- [介绍](#)
- [ESP32-C6-4MB 系列](#)
- [订阅 AT 版本发布](#)
- [AT 发布固件差异](#)

2.1.1 介绍

备注:

- 在使用下面的 AT 固件前，请仔细阅读[特别声明](#)，并遵循其中的各项条款和注意事项。
 - 如果你不确定自己的模组是否可以使用 AT 默认固件，请先阅读[AT 发布固件差异](#) 章节，该章节介绍了 ESP32-C6 AT 固件支持的命令集、硬件配置和支持的模组，帮助你确认你的模组硬件配置是否适合使用 AT 默认固件。
 - 如果你想要修改 AT 固件中下面的配置，你可以通过[at.py 工具](#) 修改 AT 固件并生成新的 AT 固件。
 - [修改 UART 配置](#)
 - [修改 Wi-Fi 配置](#)
 - [修改证书和密钥配置](#)
 - [修改 GATTS 配置](#)
-

2.1.2 ESP32-C6-4MB 系列

提示:

- 如需测试，请从[GitHub Actions](#) 里下载 esp32c6-4MB-at 固件，或者[本地编译 ESP-AT 工程](#)。
- 如需量产，请点击下面已发布的版本，填写 AT 固件申请表，获取适用于 ESP32-C6 系列芯片的 AT 固件。

已发布的版本：

- [v4.2.0.0](#) (推荐)
- [v4.1.1.0 ESP32-C6-4MB-AT-V4.1.1.0.zip](#)
- [v4.1.0.0 ESP32-C6-4MB-AT-V4.1.0.0.zip](#)
- [v4.0.0.0 ESP32-C6-4MB-AT-V4.0.0.0.zip](#)

2.1.3 订阅 AT 版本发布

请参考[订阅 AT 版本发布](#) 文档订阅我们的版本发布通知，及时获取最新版本的发布情况。

2.1.4 AT 发布固件差异

本章节比较了以下 ESP32-C6 系列的 AT 发布固件在支持的命令集、硬件、模组方面的差异。

- ESP32-C6-4MB-AT-Vx.x.x.x.zip (本节简称为 **ESP32-C6-4MB Bin**)

支持的命令集

下表列出了官方适配的 ESP32-C6 系列 AT 固件默认支持哪些命令集（用  表示）、默认不支持但可以在配置和编译 ESP-AT 工程后支持的命令集（用  表示）、完全不支持的命令集（用  表示），下表没有列出的命令集也为完全不支持的命令集。正式发布的固件见[AT 固件](#)，已适配但未发布的模组固件，需要自行编译。自行编译的固件无法从乐鑫官方服务器进行 OTA 升级。

命令集	ESP32-C6-4MB Bin
base	
user	
Wi-Fi	
TCP-IP	
mDNS	
WPS	
SmartConfig	
ping	
MQTT	
HTTP	
Bluetooth LE	
BluFi	
FileSystem	
driver	
Wi-Fi Enterprise	
Web server	
WebSocket	
OTA	

硬件差异

硬件	ESP32-C6-4MB Bin
Flash	4 MB
PSRAM	✗
UART 管脚 ¹	TX: 7 RX: 6 CTS: 5 RTS: 4

支持的模组

下表列出了官方发布的 ESP32-C6 系列 AT 固件默认支持哪些模组或芯片（用 ✓ 表示）、默认不支持但可以通过 [at.py 工具](#) 修改后支持的模组（用 ✓ 表示），以及完全不支持的模组（用 ✗ 表示）。对于完全不支持的模组，你可以 [本地编译 ESP-AT 工程](#) 修改你需要的配置后支持。

模组/芯片	ESP32-C6-4MB Bin
ESP32-C6-MINI-1	✓
ESP32-C6-WROOM-01	✓

本文档包含以下小节：

- [下载 ESP32-C6 AT 发布版固件](#)
- [AT 固件简介](#)：AT 固件包含哪些二进制文件及其作用

备注：若需下载其他芯片系列的发布版固件，请在页面左上方的下拉菜单栏选择相应的芯片，即可跳转至该芯片的文档进行下载。

AT 暂无计划支持 ESP32-H 系列、ESP32-P 系列、ESP32-S3 系列芯片，也暂无对应的 AT 固件。推荐使用 ESP32-C 系列芯片。

2.2 AT 固件简介

ESP-AT 固件包含了若干个特定功能的二进制文件：

```
build
├─ at_customize.bin      // 二级分区表（用户分区表，列出了 mfg_nvs 分区以及 fs_
└─ storage 分区的起始地址和分区大小）
├─ bootloader
│   └─ bootloader.bin    // 启动加载器
├─ customized_partitions
│   └─ mfg_nvs.bin       // 出厂配置参数，参数值见同级目录下的 mfg_nvs.csv
├─ esp-at.bin           // AT 应用固件
├─ factory
│   └─ factory_xxx.bin   //
```

(续下页)

¹ UART 管脚可自定义，详情请参考[如何设置 AT 端口管脚](#)。

(接上页)

```
→ 特定功能的二进制文件合集，你可以仅烧录本文件到起始地址为 0 的 flash_
→ 空间中，或者根据 download.config 文件中的信息将若干个二进制文件烧录到 flash_
→ 中对应起始地址的空间中。
├─ partition_table
│   └─ partition-table.bin // 一级分区表（系统分区表）
└─ ota_data_initial.bin    // OTA 数据初始化文件
```

Chapter 3

AT 命令集

本章将具体介绍如何使用各类 AT 命令。

3.1 基础 AT 命令集

- [介绍](#)
- [AT](#): 测试 AT 启动
- [AT+RST](#): 重启模块
- [AT+GMR](#): 查看版本信息
- [AT+CMD](#): 查询当前固件支持的所有命令及命令类型
- [AT+GSLP](#): 进入 Deep-sleep 模式
- [ATE](#): 开启或关闭 AT 回显功能
- [AT+RESTORE](#): 恢复出厂设置
- [AT+SAVETRANSLINK](#): 设置开机 Network/Bluetooth LE [透传模式](#) 信息
- [AT+TRANSINTVL](#): 设置[透传模式](#) 模式下的数据发送间隔
- [AT+UART_CUR](#): 设置 UART 当前临时配置，不保存到 flash
- [AT+UART_DEF](#): 设置 UART 默认配置，保存到 flash
- [AT+SLEEP](#): 设置睡眠模式
- [AT+SYSRAM](#): 查询堆空间使用情况
- [AT+SYSMSG](#): 查询/设置系统提示信息
- [AT+SYSMSGFILTER](#): 启用或禁用[系统消息](#) 过滤
- [AT+SYSMSGFILTERCFG](#): 查询/配置[系统消息](#) 的过滤器
- [AT+SYSFLASH](#): 查询或读写 flash 用户分区
- [AT+SYSMFG](#): 查询或读写[manufacturing nvs](#) 用户分区
- [AT+RFPOWER](#): 查询/设置 RF TX Power
- [AT+RFCAL](#): RF 全面校准
- [AT+SYSROLLBACK](#): 回滚到以前的固件
- [AT+SYSTIMESTAMP](#): 查询/设置本地时间戳
- [AT+SYSLOG](#): 启用或禁用 AT 错误代码提示
- [AT+SLEEPWKCFCFG](#): 设置 Light-sleep 唤醒源和唤醒 GPIO
- [AT+SYSSTORE](#): 设置参数存储模式
- [AT+SYSREG](#): 读写寄存器

3.1.1 介绍

重要： 默认的 AT 固件支持此页面下的所有 AT 命令。

3.1.2 AT：测试 AT 启动

执行命令

命令：

```
AT
```

响应：

```
OK
```

3.1.3 AT+RST：重启模块

执行命令

命令：

```
AT+RST
```

响应：

```
OK
```

3.1.4 AT+GMR：查看版本信息

执行命令

命令：

```
AT+GMR
```

响应：

```
<AT version info>
<SDK version info>
<compile time>
Bin version:<Bin version>(<module_name>)

OK
```

参数

- **<AT version info>**：AT 核心库的版本信息，它们在 `esp-at/components/at/lib/` 目录下。代码是闭源的，无开放计划。
- **<SDK version info>**：AT 使用的平台 SDK 版本信息，它们定义在 `esp-at/module_config/module_{platform}_default/IDF_VERSION` 文件中。
- **<compile time>**：固件生成时间。

- **<Bin version>**: AT 固件版本信息。版本信息可以在 `menuconfig` 中修改, `python build.py menuconfig>Application manager>Project version`。最大长度: 32 字节。
- **<module_name>**: 模块名称, 在 `esp-at/components/customized_partitions/raw_data/factory_param/factory_param_data.csv` 里定义。

说明

- 如果你在使用 ESP-AT 固件中有任何问题, 请先提供 AT+GMR 版本信息。

示例

```
AT+GMR
AT version:2.2.0.0-dev(ca41ec4 - ESP32-C6 - Sep 16 2020 11:28:17)
SDK version:v4.0.1-193-ge7ac221b4
compile time(98b95fc):Oct 29 2020 11:23:25
Bin version:2.1.0(MINI-1)

OK
```

3.1.5 AT+CMD: 查询当前固件支持的所有命令及命令类型

查询命令

命令:

```
AT+CMD?
```

响应:

```
+CMD:<index>,<"AT command name">,<support test command>,<support query command>,<support set command>,<support execute command>

OK
```

参数

- **<index>**: AT 命令序号
- **<" AT command name" >**: AT 命令名称
- **<support test command>**: 0 表示不支持, 1 表示支持
- **<support query command>**: 0 表示不支持, 1 表示支持
- **<support set command>**: 0 表示不支持, 1 表示支持
- **<support execute command>**: 0 表示不支持, 1 表示支持

3.1.6 AT+GSLP: 进入 Deep-sleep 模式

设置命令

命令:

```
AT+GSLP=<time>
```

响应:

```
<time>
```

```
OK
```

参数

- **<time>**: 设备进入 Deep-sleep 的时长，单位：毫秒。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。
 - 最小 Deep-sleep 时长为 5 毫秒。
 - 最大 Deep-sleep 时长约为 28.8 天 ($2^{31}-1$ 毫秒)。

说明

- 由于外部因素的影响，所有设备进入 Deep-sleep 的实际时长与理论时长之间会存在差异。

3.1.7 ATE：开启或关闭 AT 回显功能

执行命令

命令：

```
ATE0
```

或

```
ATE1
```

响应：

```
OK
```

参数

- **ATE0**: 关闭回显
- **ATE1**: 开启回显

3.1.8 AT+RESTORE：恢复出厂设置

执行命令

命令：

```
AT+RESTORE
```

响应：

```
OK
```

说明

- 该命令将擦除所有保存到 flash 的参数，并恢复为默认参数。
- 运行该命令会重启设备。

3.1.9 AT+SAVETRANSLINK: 设置开机 Network/Bluetooth LE 透传模式信息

- 设置开机进入 *TCP/SSL* 客户端透传模式信息
- 设置开机进入 *TCP/SSL* 服务器透传模式信息
- 设置开机进入 *UDP* 透传模式信息
- 设置开机进入 *BLE* 透传模式信息

设置开机进入 TCP/SSL 客户端透传模式信息

设置命令 命令:

```
AT+SAVETRANSLINK=<mode>,<"remote host">,<remote port>[,<"type">,<keep_alive>]
```

响应:

```
OK
```

参数

- **<mode>**:
 - 0: 关闭 ESP32-C6 上电进入 TCP/SSL 客户端透传模式
 - 1: 开启 ESP32-C6 上电进入 TCP/SSL 客户端透传模式
- **<" remote host" >**: 字符串参数, 表示远端 IPv4 地址、IPv6 地址, 或域名。最长为 64 字节。
- **<remote port>**: 远端端口值。
- **<" type" >**: 字符串参数, 表示传输类型: "TCP", "TCPv6", "SSL", 或 "SSLv6"。默认值: "TCP"。
- **<keep_alive>**: 配置套接字的 SO_KEEPALIVE 选项 (参考: [SO_KEEPALIVE 介绍](#)), 单位: 秒。
 - 范围: [0,7200]。
 - * 0: 禁用 keep-alive 功能 (默认)。
 - * 1 ~ 7200: 开启 keep-alive 功能。TCP_KEEPIDLE 值为 <keep_alive>, TCP_KEEPINTVL 值为 1, TCP_KEEPCNT 值为 3。
 - 本命令中的 <keep_alive> 参数与 AT+CIPTCPPOPT 命令中的 <keep_alive> 参数相同, 最终值由后设置的命令决定。如果运行本命令时不设置 <keep_alive> 参数, 则默认使用上次配置的值。

说明

- 本设置将开机透传模式信息保存在 NVS 区, 若参数 <mode> 为 1, 下次上电自动进入透传模式。需重启生效。

示例

```
AT+SAVETRANSLINK=1,"192.168.6.110",1002,"TCP"
AT+SAVETRANSLINK=1,"www.baidu.com",443,"SSL"
AT+SAVETRANSLINK=1,"240e:3a1:2070:11c0:55ce:4e19:9649:b75",8080,"TCPv6"
AT+SAVETRANSLINK=1,"240e:3a1:2070:11c0:55ce:4e19:9649:b75",8080,"SSLv6"
```

设置开机进入 TCP/SSL 服务器透传模式信息

设置 命令:

```
AT+SAVETRANSLINK=<mode>[,<port>][,<"type">][,<CA enable>][,<netif>]
```

响应:

OK

参数

- **<mode>**:
 - 0: 关闭 ESP32-C6 上电进入 TCP/SSL 服务器透传模式
 - 3: 开启 ESP32-C6 上电进入 TCP/SSL 服务器透传模式
- **<port>**: 服务器监听的端口, 默认值: 333。
- **<" type" >**: 服务器类型:" TCP", " TCPv6", " SSL", 或 "SSLv6"。默认值:" TCP"。
- **<CA enable>**: 仅在 <type> 为 "SSL" 或 "SSLv6" 时才有此参数。
 - 0: 不使用 CA 认证
 - 1: 使用 CA 认证
- **<netif>**: 指定服务器监听的网络接口, 默认值: 0。
 - 0: 所有网络接口
 - 1: Wi-Fi station 接口
 - 2: Wi-Fi SoftAP 接口

说明

- 本设置将开机透传模式信息保存在 NVS 区, 若参数 <mode> 为 3, 下次上电自动进入透传模式。需重启生效。
- ESP32-C6 作为 Wi-Fi Station 时, 请提前配置好 Wi-Fi 连接信息并启用重连机制; 作为 SoftAP 时, 请提前配置好 SoftAP 参数, 保证上电后能正常工作。
- 在 TCP/SSL 服务器的透传模式下, 仅支持单一透传连接: 一旦有客户端建立透传连接, 服务器将进入透传模式并拒绝后续的透传连接请求。
- 当底层网络接口 (netif) 停止或断开时, 已有连接会被关闭, 服务器会恢复为监听状态, 准备接受新的普通连接。
- 退出透传模式后, TCP/SSL 服务器仍然保持运行, 但以普通 (非透传) 模式提供服务, 最大并发连接数为 1。

示例

```
// 设置 Wi-Fi SoftAP 信息, 名称为 esp-ap-001, 密码为空, 信道 11, 加密方式为开放, 最大连接数 3
AT+CWSAP="esp-ap-001","",11,0,3

// 设置开机进入 TCP 服务器透传模式, 监听端口 1002
AT+SAVETRANSLINK=3,1002,"TCP",2

// 重启模块
AT+RST

// PC 连接到 esp-ap-001, 并通过 TCP 客户端工具连接到 ESP32-C6 的 1002 端口
nc 192.168.4.1 1002

// 透传数据
```

设置开机进入 UDP 透传模式信息

设置 命令:

```
AT+SAVETRANSLINK=<mode>,<"remote host">,<remote port>,<["type"]>,<local port>]
```

响应:

OK

参数

- **<mode>**:
 - 0: 关闭 ESP32-C6 上电进入 UDP 透传模式
 - 1: 开启 ESP32-C6 上电进入 UDP 透传模式
- **<" remote host">**: 字符串参数, 表示远端 IPv4 地址、IPv6 地址, 或域名。最长为 64 字节。
- **<remote port>**: 远端端口值。
- **<" type">**: 字符串参数, 表示传输类型: "UDP" 或 "UDPv6"。
- **<local port>**: 开机进入 UDP 传输时, 使用的本地端口。

说明

- 本设置将 UDP 开机透传模式信息保存在 NVS 区, 若参数 <mode> 为 1, 下次上电自动进入 UDP 透传模式。需重启生效。
- 如果你想基于 IPv6 网络建立一个 UDP 传输, 请执行以下操作:
 - 确保 AP 支持 IPv6
 - 设置 `AT+CIPV6=1`
 - 通过 `AT+CWJAP` 命令获取到一个 IPv6 地址
 - (可选) 通过 `AT+CIPSTA?` 命令检查 ESP32-C6 是否获取到 IPv6 地址

示例

```
AT+SAVETRANSLINK=1,"192.168.6.110",1002,"UDP",1005
AT+SAVETRANSLINK=1,"240e:3a1:2070:11c0:55ce:4e19:9649:b75",8081,"UDPv6",1005
```

设置开机进入 BLE 透传模式信息

设置 命令:

```
AT+SAVETRANSLINK=<mode>,<role>,<tx_srv>,<tx_char>,<rx_srv>,<rx_char>,<"peer_addr">
```

响应:

OK

参数

- **<mode>**:
 - 0: 关闭 ESP32-C6 上电进入 BLE 透传模式
 - 2: 开启 ESP32-C6 上电进入 BLE 透传模式
- **<role>**:
 - 1: client 角色
 - 2: server 角色
- **<tx_srv>**: tx 服务序号。AT 作为 GATTTC 时, 通过 `AT+BLEGATTCPRIMSRV=<conn_index>` 命令查询; 作为 GATTTS 时, 通过 `AT+BLEGATTSSRV?` 命令查询。
- **<tx_char>**: tx 服务特征序号。AT 作为 GATTTC 时, 通过 `AT+BLEGATTCCCHAR=<conn_index>,<srv_index>` 命令查询; 作为 GATTTS 时, 通过 `AT+BLEGATTSCCHAR?` 命令查询。
- **<rx_srv>**: rx 服务序号。AT 作为 GATTTC 时, 通过 `AT+BLEGATTCPRIMSRV=<conn_index>` 命令查询; 作为 GATTTS 时, 通过 `AT+BLEGATTSSRV?` 命令查询。
- **<rx_char>**: rx 服务特征序号。AT 作为 GATTTC 时, 通过 `AT+BLEGATTCCCHAR=<conn_index>,<srv_index>` 命令查询; 作为 GATTTS 时, 通过 `AT+BLEGATTSCCHAR?` 命令查询。
- **<" peer_addr">**: 对方 Bluetooth LE 地址。

说明

- 本设置将 BLE 开机透传模式信息保存在 NVS 区, 若参数 <mode> 为 2, 下次上电自动进入 Bluetooth LE 透传模式。需重启生效。

示例

```
AT+SAVETRANSLINK=2,2,1,7,1,5,"26:a2:11:22:33:88"
```

3.1.10 AT+TRANSINTVL: 设置透传模式模式下的数据发送间隔

查询命令

命令:

```
AT+TRANSINTVL?
```

响应:

```
+TRANSINTVL:<interval>
```

```
OK
```

设置命令

命令:

```
AT+TRANSINTVL=<interval>
```

响应:

```
OK
```

参数

- <interval>: 数据发送间隔。单位: 毫秒。默认值: 20。范围: [0,1000]。

说明

- 透传模式下, 当 ESP32-C6 从 UART 接收到数据后, 如果收到的数据长度大于等于 2920 字节, 数据会立即被分为每 2920 字节一组的块进行发送, 否则会等待 <interval> 毫秒或等待收到的数据大于等于 2920 字节再发送数据。
- 当数据量很小, 且数据发送间隔很短时, 可以通过设置 <interval> 来调整数据发送的时机。当 <interval> 很小时, 可以降低向协议栈发送数据的延时, 但这会增加协议栈数据向网络发送的次数, 一定程度降低了吞吐性能。

示例

```
// 设置收到数据后立即发送
AT+TRANSINTVL=0
```

3.1.11 AT+UART_CUR: 设置 UART 当前临时配置, 不保存到 flash

查询命令

命令:

```
AT+UART_CUR?
```

响应:

```
+UART_CUR:<baudrate>,<databits>,<stopbits>,<parity>,<flow control>
```

```
OK
```

设置命令

命令:

```
AT+UART_CUR=<baudrate>,<databits>,<stopbits>,<parity>,<flow control>
```

响应:

```
OK
```

参数

- **<baudrate>**: UART 波特率
 - ESP32-C6 设备: 支持范围为 80 ~ 5000000
- **<databits>**: 数据位
 - 5: 5 bit 数据位
 - 6: 6 bit 数据位
 - 7: 7 bit 数据位
 - 8: 8 bit 数据位
- **<stopbits>**: 停止位
 - 1: 1 bit 停止位
 - 2: 1.5 bit 停止位
 - 3: 2 bit 停止位
- **<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
 - 3: 同时使能 RTS 和 CTS

说明

- 查询命令返回的是 UART 配置参数的实际值, 由于时钟分频的原因, 可能与设定值有细微的差异。
- 本设置不保存到 flash。
- 使用硬件流控功能需要连接设备的 CTS/RTS 管脚, 详情请见[硬件连接](#)和 components/customized_partitions/raw_data/factory_param/factory_param_data.csv。

示例

```
AT+UART_CUR=115200,8,1,0,3
```

3.1.12 AT+UART_DEF: 设置 UART 默认配置，保存到 flash

查询命令

命令:

```
AT+UART_DEF?
```

响应:

```
+UART_DEF:<baudrate>,<databits>,<stopbits>,<parity>,<flow control>
```

```
OK
```

设置命令

命令:

```
AT+UART_DEF=<baudrate>,<databits>,<stopbits>,<parity>,<flow control>
```

响应:

```
OK
```

参数

- **<baudrate>**: UART 波特率
 - ESP32-C6 设备: 支持范围为 80 ~ 5000000
- **<databits>**: 数据位
 - 5: 5 bit 数据位
 - 6: 6 bit 数据位
 - 7: 7 bit 数据位
 - 8: 8 bit 数据位
- **<stopbits>**: 停止位
 - 1: 1 bit 停止位
 - 2: 1.5 bit 停止位
 - 3: 2 bit 停止位
- **<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
 - 3: 同时使能 RTS 和 CTS

说明

- 配置更改将保存在 NVS 分区，当设备再次上电时仍然有效。

- 使用硬件流控功能需要连接设备的 CTS/RTS 管脚，详情请见[硬件连接](#) 和 components/customized_partitions/raw_data/factory_param/factory_param_data.csv。

示例

```
AT+UART_DEF=115200,8,1,0,3
```

3.1.13 AT+SLEEP：设置睡眠模式

查询命令

命令：

```
AT+SLEEP?
```

响应：

```
+SLEEP:<sleep mode>
OK
```

设置命令

命令：

```
AT+SLEEP=<sleep mode>
```

响应：

```
OK
```

参数

- **<sleep mode>**：
 - 0: 禁用睡眠模式。
 - 1: Wi-Fi Modem-sleep 模式。射频模块将根据 AP 的 DTIM 定期关闭。仅在 Wi-Fi 模式为 Station 时设置生效；在无 Wi-Fi 模式下，允许设置，但不会生效（为了兼容旧版 AT 固件）；其他情况下不可设置。
 - 2: Light-sleep 模式。在 Wi-Fi 模式为 SoftAP 或 Station+SoftAP 时不可设置。
 - * 无 Wi-Fi 模式
 - CPU 将自动进入睡眠，射频模块将关闭。
 - * 单 Wi-Fi 模式
 - CPU 将自动进入睡眠，射频模块也将根据 [AT+CWJAP](#) 或 [AT+CWCONFIG](#) 命令设置的 listen interval 参数定期关闭。
 - * 单 Bluetooth 模式
 - 在 Bluetooth 广播态下，CPU 将自动进入睡眠，射频模块也将根据广播间隔定期关闭。
 - 在 Bluetooth 连接态下，CPU 将自动进入睡眠，射频模块也将根据连接间隔定期关闭。
 - * Wi-Fi 和 Bluetooth 共存模式
 - CPU 将自动进入睡眠，射频模块根据电源管理模块定期关闭。
 - 3: Wi-Fi Modem-sleep listen interval 模式。射频模块将根据 [AT+CWJAP](#) 或 [AT+CWCONFIG](#) 命令设置的 listen interval 参数定期关闭。仅在 Wi-Fi 模式为 Station 时设置生效；在无 Wi-Fi 模式下，允许设置，但不会生效（为了兼容旧版 AT 固件）；其他情况下不可设置。

说明

- 设置 Light-sleep 模式前，建议提前通过 [AT+SLEEPWKCFCG](#) 命令设置好唤醒源，否则没法唤醒，设备将一直处于睡眠状态。
- 设置 Light-sleep 模式后，如果 Light-sleep 唤醒条件不满足时，设备将自动进入睡眠模式，当 Light-sleep 唤醒条件满足时，设备将自动从睡眠模式中唤醒。
- 单 BLE 模式下，只有 BLE Light-sleep 和 BLE Modem-sleep 两种睡眠模式。通过 AT+SLEEP=2 命令启用 BLE Light-sleep，通过 AT+SLEEP=0 命令启用 BLE Modem-sleep。
- 对于 BLE 模式下的 Light-sleep 模式，用户必须确保外接 32KHz 晶振，否则，Light-sleep 模式会以 Modem-sleep 模式工作。
- AT+SLEEP 更多示例请参考文档 [Sleep AT 示例](#)。

示例

```
AT+SLEEP=0
```

3.1.14 AT+SYSRAM: 查询堆空间使用情况

查询命令

命令:

```
AT+SYSRAM?
```

响应:

```
+SYSRAM:<remaining RAM size>,<minimum heap size>
OK
```

参数

- **<remaining RAM size>**: 当前可用的堆空间大小，单位: byte。当有内存泄漏时，该值会逐渐减小。
- **<minimum heap size>**: ESP32-C6 上电以来记录到的最小可用堆空间大小，单位: byte。当 <minimum heap size> 小于或接近于 10 KB 时，可能影响 ESP32-C6 的 Wi-Fi 和低功耗蓝牙功能。此数值仅会在出现新低值时更新，不会增加。

说明

- 当 <minimum heap size> 小于或接近 10 KB，或在 [AT 日志端口](#) 看到类似 alloc failed, size:<requested_size>, caps:<requested_caps> 的日志时，请考虑进行内存优化：
 - 在 python build.py menuconfig>Component config>AT 路径下禁用不需要的 AT 功能。
 - 参考 [内存优化](#) 文档中的相关方法。

示例

```
AT+SYSRAM?
+SYSRAM:148408,84044
OK
```

设置命令

功能:

查询在给定能力下的系统内存使用情况。

命令:

```
AT+SYSRAM=<caps>
```

响应:

```
+SYSRAM:<caps_largest_free_block_size>,<caps_free_size>,<caps_minimum_free_size>,<caps_total_size>
OK
```

参数

- **<caps>**: 能力值。详见 [不同 capabilities 定义](#)。可以组合使用，如 AT+SYSRAM=0x1800，表示 `MALLOC_CAP_INTERNAL | MALLOC_CAP_DEFAULT`。
- **<caps_largest_free_block_size>**: 当前 caps 下能够分配的最大空闲块，单位：byte。
- **<caps_free_size>**: 当前 caps 下所有的空闲块大小总和，单位：byte。
- **<caps_minimum_free_size>**: 芯片上电后，caps 下所有的空闲块大小总和的最小值，单位：byte。
- **<caps_total_size>**: 当前 caps 下总内存大小，单位：byte。

说明

- 系统运行过程中，一旦内存不足，**AT 日志端口** 会输出 `alloc failed, size:<requested_size>, caps:<requested_caps>`。你可以发送 `AT+SYSRAM=<requested_caps>` 查看当前 caps 的内存使用情况。其中 `<caps_largest_free_block_size>` 决定了能否分配 `<requested_size>` 大小的内存块。

3.1.15 AT+SYMSG: 查询/设置系统提示信息

查询命令

功能:

查询当前系统提示信息状态

命令:

```
AT+SYMSG?
```

响应:

```
+SYMSG:<state>
OK
```

设置命令

功能:

设置系统提示信息。如果你需要更加精细的管理 AT 消息，请使用 [AT+SYMSGFILTER](#) 命令。

命令:

```
AT+SYSMSG=<state>
```

响应:

```
OK
```

参数

• <state>:

- bit 0: 退出 Network 透传模式, Bluetooth LE SPP 及 Bluetooth SPP 时是否打印提示信息
 - * 0: 不打印
 - * 1: 打印 +QUIT
- bit 1: 连接时提示信息类型
 - * 0: 使用简单版提示信息, 如 XX, CONNECT
 - * 1: 使用详细版提示信息, 如 +LINK_CONN:status_type,link_id,ip_type,terminal_type,remote_ip,remote_port,local_port
- bit 2: 连接状态提示信息, 适用于 Network 透传模式、Bluetooth LE SPP 及 Bluetooth SPP
 - * 0: 不打印提示信息
 - * 1: 当 Wi-Fi、socket、Bluetooth LE 或 Bluetooth 状态发生改变时, 打印提示信息, 如:

```
- "CONNECT\r\n" 或以 "+LINK_CONN:" 开头的提示信息
- "CLOSED\r\n"
- "WIFI CONNECTED\r\n"
- "WIFI GOT IP\r\n"
- "WIFI GOT IPv6 LL\r\n"
- "WIFI GOT IPv6 GL\r\n"
- "WIFI DISCONNECT\r\n"
- "+ETH_CONNECTED\r\n"
- "+ETH_DISCONNECTED\r\n"
- 以 "+ETH_GOT_IP:" 开头的提示信息
- 以 "+STA_CONNECTED:" 开头的提示信息
- 以 "+STA_DISCONNECTED:" 开头的提示信息
- 以 "+DIST_STA_IP:" 开头的提示信息
- 以 "+BLECONN:" 开头的提示信息
- 以 "+BLEDISCONN:" 开头的提示信息
- "+BLE_TT_MODE\r\n"
- 在透传模式下, 使能 notify 或者 indicate 时, 打印以 "+WRITE:" 开头的提示信息
```

说明

- 若 `AT+SYSTORE=1`, 配置更改将被保存在 NVS 分区。
- 若设 bit 0 为 1, 退出透传模式时会提示 +QUIT。
- 若设 bit 1 为 1, 将会影响 `AT+CIPSTART` 和 `AT+CIPSERVER` 命令, 系统将提示 “+LINK_CONN:status_type,link_id,ip_type,terminal_type,remote_ip,remote_port,local_port”, 而不是 “XX,CONNECT”。

示例

```
// 退出透传模式时不打印提示信息
// 连接时打印详细版提示信息
// 连接状态发生改变时不打印信息
AT+SYSMSG=2
```

或

```
// 透传模式下，Wi-Fi、socket、Bluetooth LE 或 Bluetooth 状态改变时会打印提示信息
AT+SYMSMSG=4
```

3.1.16 AT+SYMSMSGFILTER：启用或禁用系统消息过滤

查询命令

功能：

查询当前系统信息过滤的状态

命令：

```
AT+SYMSMSGFILTER?
```

响应：

```
+SYMSMSGFILTER:<enable>
```

```
OK
```

设置命令

功能：

启用或禁用系统消息过滤

命令：

```
AT+SYMSMSGFILTER=<enable>
```

响应：

```
OK
```

参数

- **<enable>**:
 - 0: 禁用系统消息过滤。系统默认值。禁用后，系统消息不会被设置的过滤器过滤。
 - 1: 启用系统消息过滤。开启后，系统消息被正则表达式匹配上的数据会被 AT 过滤掉，MCU 不会收到；而未被正则表达式匹配上的数据，会原样发往 MCU。

说明

- 请先使用 **AT+SYMSMSGFILTERCFG** 命令配置有效的过滤器，再通过本命令启用或禁用系统消息过滤，实现更加精细的系统消息管理。
- 请谨慎使用 **AT+SYMSMSGFILTER=1** 命令，建议你开启系统消息过滤后要及时禁用，防止 AT 的系统消息被过度过滤。
- 在进入**透传模式**前，强烈建议使用 **AT+SYMSMSGFILTER=0** 命令，禁用系统消息过滤。
- 如果你基于 AT 工程二次开发，请使用如下的 APIs 实现 AT 命令口的数据发送。

```
// 原生的 AT 命令口的数据发送。数据不会被 AT+SYMSMSGFILTER_
↪命令过滤，发送数据前也不会唤醒 MCU（AT+USERWKMCCCFG 命令设置的 MCU 唤醒功能）。
int32_t esp_at_port_write_data_without_filter(uint8_t data, int32_t len);
```

```
// 具有过滤功能的 AT 命令口的数据发送。数据会被 AT+SYMSMSGFILTER_
```

(续下页)

(接上页)

```

↪命令过滤（如果启用），发送数据前不会唤醒 MCU（AT+USERWKMUCUCFG 命令设置的 MCU_
↪唤醒功能）。
int32_t esp_at_port_write_data(uint8_t data, int32_t len);

// 具有唤醒 MCU 功能的 AT 命令口的数据发送。数据不会被 AT+SYSMSGFILTER_
↪命令过滤，发送数据前会唤醒 MCU（AT+USERWKMUCUCFG 命令设置的 MCU 唤醒功能）。
int32_t esp_at_port_active_write_data_without_filter(uint8_t data, int32_t len);

// 同时具有唤醒 MCU 功能和过滤功能的 AT 命令口的数据发送。数据会被 AT+SYSMSGFILTER_
↪命令过滤（如果启用），发送数据前会唤醒 MCU（AT+USERWKMUCUCFG 命令设置的 MCU_
↪唤醒功能）。
int32_t esp_at_port_active_write_data(uint8_t data, int32_t len);

```

示例 详细示例参考：[系统消息过滤示例](#)。

3.1.17 AT+SYSMSGFILTERCFG：查询/配置系统消息的过滤器

- [查询过滤器](#)
- [清除所有过滤器](#)
- [增加一个过滤器](#)
- [删除一个过滤器](#)

查询过滤器

查询命令 命令：

```
AT+SYSMSGFILTERCFG?
```

响应：

```
+SYSMSGFILTERCFG:<index>,<"head_regexp">,<"tail_regexp">
OK
```

参数

- **<index>**：过滤器的索引。
- **<" head_regexp" >**：头部正则表达式。
- **<" tail_regexp" >**：尾部正则表达式。

清除所有过滤器

设置命令 命令：

```
AT+SYSMSGFILTERCFG=<operator>
```

响应：

```
OK
```

参数

- **<operator>**:
 - 0: 清除所有过滤器。清除后，可以释放一些过滤器所占用的堆空间大小。

示例

```
// 清除所有过滤器
AT+SYMSGFILTERCFG=0
```

增加一个过滤器

设置命令 命令：

```
AT+SYMSGFILTERCFG=<operator>,<head_regexp_len>,<tail_regexp_len>[,<cflags>]
```

响应：

```
OK
```

```
>
```

上述响应表示 AT 已准备好接收 AT 命令口的数据，此时你可以输入数据（即：头部正则表达式和尾部正则表达式），当 AT 接收到的数据长度达到 `<head_regexp_len> + <tail_regexp_len>` 后，进行正则表达式完整性校验。

如果正则表达式完整性校验失败或添加过滤器失败，返回：

```
ERROR
```

如果正则表达式完整性校验成功且添加过滤器成功，返回：

```
OK
```

参数

- **<operator>**：
 - 1: 增加一个过滤器。一个过滤器包含头部正则表达式和尾部正则表达式。
- **<head_regexp_len>**：头部正则表达式长度。范围：[0,64]。如果设置为 0，代表忽略头部正则表达式的匹配，同时 `<tail_regexp_len>` 不能为 0。
- **<tail_regexp_len>**：尾部正则表达式长度。范围：[0,64]。如果设置为 0，代表忽略尾部正则表达式的匹配，同时 `<head_regexp_len>` 不能为 0。
- **<cflags>**：可选参数，正则表达式的编译标志，详见 [cflags 说明](#)。默认值：0。
 - bit 0: REG_EXTENDED，使用 POSIX 扩展正则表达式语法。
 - bit 1: REG_ICASE，忽略大小写匹配。
 - bit 2: REG_NEWLINE，改变 ^ 和 \$ 的行为，使其匹配行的开头和结尾，而不是整个字符串的开头和结尾。

说明

- 请先使用本命令配置有效的过滤器，再通过 `AT+SYMSGFILTER` 命令启用或禁用系统消息过滤，实现更加精细的系统消息管理。
- 头部和尾部正则表达式格式参考 [POSIX 基本正则语法 \(BRE\)](#)。
- 为了避免系统消息（AT 命令口的 TX 数据）被错误过滤，**强烈建议**头部正则表达式以 ^ 开头，尾部正则表达式以 \$ 结束。
- 只有系统消息 **同时匹配**上头部正则表达式和尾部正则表达式时，系统消息才会被过滤。过滤后，系统消息被正则表达式匹配上的数据会被 AT 过滤掉，MCU 不会收到；而未被正则表达式匹配上的数据，会原样发往 MCU。例如：收到数据 data，长度为 n 字节，头部正则表达式匹配了 data[0] ~ data[i]，尾部正则表达式匹配了 data[j] ~ data[n-1]，那么最终 MCU 收到的数据为 data[i+1] ~ data[j-1]。
- 当系统消息匹配到一个过滤器后，不会继续匹配其它的过滤器。
- 系统消息匹配过滤器时，系统消息不会缓存，即不会将上一条的系统消息和本条系统消息合在一起，进行匹配。
- 对于吞吐量较大的设备，强烈建议你设置较少的过滤器，同时及时通过 `AT+SYMSGFILTER=0` 命令禁用系统消息过滤。

示例

```
// 设置过滤器，过滤掉 "WIFI CONNECTED" 系统消息报告
AT+SYMSGFILTERCFG=1,17,0
// 等待命令返回 OK 和 > 后，输入 ^WIFI CONNECTED\r\n（注意 \r\n 占用 2 个字节，对应 ASCII 码中的 0D 0A）

// 开启系统消息过滤
AT+SYMSGFILTER=1

// 测试过滤功能
AT+CWMODE=1
AT+CWJAP="ssid","password"
// AT 不再输出 WIFI CONNECTED 系统消息报告
```

详细示例参考：[系统消息过滤示例](#)。

删除一个过滤器

设置命令 命令：

```
AT+SYMSGFILTERCFG=<operator>,<head_regexp_len>,<tail_regexp_len>[,<cflags>]
```

响应：

```
OK
```

```
>
```

上述响应表示 AT 已准备好接收 AT 命令口的数据，此时你可以输入数据（即：头部正则表达式和尾部正则表达式），当 AT 接收到的数据长度达到 <head_regexp_len> + <tail_regexp_len> 后，进行正则表达式完整性校验。

如果正则表达式完整性校验失败或删除过滤器失败，返回：

```
ERROR
```

如果正则表达式完整性校验成功且删除过滤器成功，返回：

```
OK
```

参数

- **<operator>**：
 - 2: 删除一个过滤器。
- **<head_regexp_len>**：头部正则表达式长度。范围：[0,64]。如果设置为 0，则 <tail_regexp_len> 不能为 0。
- **<tail_regexp_len>**：尾部正则表达式长度。范围：[0,64]。如果设置为 0，则 <head_regexp_len> 不能为 0。
- **<cflags>**：可选参数，正则表达式的编译标志，详见 [cflags 说明](#)。默认值：0。
 - bit 0: REG_EXTENDED，使用 POSIX 扩展正则表达式语法。
 - bit 1: REG_ICASE，忽略大小写匹配。
 - bit 2: REG_NEWLINE，改变 ^ 和 \$ 的行为，使其匹配行的开头和结尾，而不是整个字符串的开头和结尾。

说明

- 待删除的过滤器应在已增加的过滤器中。

示例

```
// 删除上述添加的过滤器
AT+SYMSGFILTERCFG=2,17,0
// 等待命令返回 OK 和 > 后, 输入 ^WIFI CONNECTED\r\n (注意 \r\n 占用 2
↳个字节, 对应 ASCII 码中的 0D 0A)

// 测试功能
AT+CWMODE=1
AT+CWJAP="ssid","password"
// AT 会输出 WIFI CONNECTED 系统消息报告
```

3.1.18 AT+SYSFLASH: 查询或读写 flash 用户分区**查询命令****功能:**

查询 flash 用户分区

命令:

```
AT+SYSFLASH?
```

响应:

```
+SYSFLASH:<partition>,<type>,<subtype>,<addr>,<size>
OK
```

设置命令**功能:**

读、写、擦除 flash 用户分区

命令:

```
AT+SYSFLASH=<operation>,<partition>,<offset>,<length>
```

响应:

```
+SYSFLASH:<length>,<data>
OK
```

参数

- **<operation>**:
 - 0: 擦除分区
 - 1: 写分区
 - 2: 读分区
- **<partition>**: 用户分区名称
- **<offset>**: 偏移地址
- **<length>**: 数据长度
- **<type>**: 用户分区类型
- **<subtype>**: 用户分区子类型
- **<addr>**: 用户分区地址
- **<size>**: 用户分区大小

说明

- 使用本命令需烧录 `at_customize.bin`，详细信息可参考[如何自定义分区](#)。
- 擦除分区时，请完整擦除该目标分区。这可以通过省略 `<offset>` 和 `<length>` 参数来完成。例如，命令 `AT+SYSFLASH=0,"mfg_nvs"` 可擦除整个“mfg_nvs”区域。
- 关于分区的定义可参考 [ESP-IDF 分区表](#)。
- 当 `<operator>` 为 `write` 时，系统收到此命令后先换行返回 `>`，此时你可以输入要写的的数据，数据长度应与 `<length>` 一致。
- 写分区前，请先擦除该分区。
- 如果你想修改 `mfg_nvs` 分区中的某些数据，请使用 `AT+SYSMFG` 命令（NVS 中的键值对操作）。如果你想修改整个 `mfg_nvs` 分区，请使用 `AT+SYSFLASH` 命令（分区操作）。
- 写分区时，MCU 应该分次写入数据，避免一次性写入过多数据导致内存不足。例如每次写入 4 KB 字节数据，直到写入完成。

示例

```
// 擦除整个 "mfg_nvs" 分区
AT+SYSFLASH=0,"mfg_nvs"

// 在 "mfg_nvs" 分区偏移地址 0 处写入新的 "mfg_nvs" 分区（大小为 0x1C000）
AT+SYSFLASH=1,"mfg_nvs",0,0x1C000
```

3.1.19 AT+SYSMFG：查询或读写 manufacturing nvs 用户分区

查询命令

功能：

查询 manufacturing nvs 用户分区内的命名空间 (namespace)

命令：

```
AT+SYSMFG?
```

响应：

```
+SYSMFG:<"namespace">
OK
```

擦除命名空间或键值对

设置命令 命令：

```
AT+SYSMFG=<operation>,<"namespace">[,<"key">]
```

响应：

```
OK
```

参数

- **<operation>**:
 - 0: 擦除操作
 - 1: 读取操作
 - 2: 写入操作

- **<" namespace">**: 命名空间。
- **<" key">**: 主键，或称为键。当 **<"key">** 缺省时，擦除 **<"namespace">** 内所有的键值对；否则只擦除当前指定的 **<"key">** 的键值对。

说明

- 请先阅读 [非易失性存储 \(NVS\)](#)，了解命名空间、键值对的概念。

示例

```
// 擦除 client_cert 命名空间内所有的键值对（即：擦除默认的第 0 套和第 1
↪套客户端证书）
AT+SYSMF0=0,"client_cert"

// 擦除 client_cert 命名空间内的 client_cert.0 键值对（即：擦除默认的第 0
↪套客户端证书）
AT+SYSMF0=0,"client_cert","client_cert.0"
```

读取命名空间或键值对

设置命令 命令：

```
AT+SYSMF0=<operation>[,<"namespace">][,<"key">][,<offset>,<length>]
```

响应：

当 **<"namespace">** 以及之后参数缺省时，返回：

```
+SYSMF0:<"namespace">
OK
```

当 **<"key">** 以及之后参数缺省时，返回：

```
+SYSMF0:<"namespace">,<"key">,<type>
OK
```

其余情况，返回：

```
+SYSMF0:<"namespace">,<"key">,<type>,<length>,<value>
OK
```

参数

- **<operation>**:
 - 0: 擦除操作
 - 1: 读取操作
 - 2: 写入操作
- **<" namespace">**: 命名空间。
- **<" key">**: 主键，或称为键。
- **<offset>**: 键值的偏移。
- **<length>**: 键值的长度。
- **<type>**: 键值的类型。
 - 1: u8
 - 2: i8
 - 3: u16

- 4: i16
- 5: u32
- 6: i32
- 7: string
- 8: binary
- **<value>**: 键值的数据。

说明

- 请先阅读 [非易失性存储 \(NVS\)](#)，了解命名空间、键值对的概念。

示例

```
// 读取当前所有的命名空间
AT+SYSMF=1

// 读取 client_cert 命名空间内所有的主键
AT+SYSMF=1,"client_cert"

// 读取 client_cert 命名空间内的 client_cert.0 主键的值
AT+SYSMF=1,"client_cert","client_cert.0"

// 读取 client_cert 命名空间内的 client_cert.0 主键的值，从偏移 100 的位置读取 200_
↪ 字节
AT+SYSMF=1,"client_cert","client_cert.0",100,200
```

向命名空间内写入键值对

设置命令 命令：

```
AT+SYSMF=<operation>,<"namespace">,<"key">,<type>,<value>
```

响应：

```
OK
```

参数

- **<operation>**:
 - 0: 擦除操作
 - 1: 读取操作
 - 2: 写入操作
- **<" namespace" >**: 命名空间。
- **<" key" >**: 主键，或称为键。
- **<type>**: 键值的类型。
 - 1: u8
 - 2: i8
 - 3: u16
 - 4: i16
 - 5: u32
 - 6: i32
 - 7: string
 - 8: binary
- **<value>**: 参数 <type> 不同，则此参数意义不同：
 - 如果 <type> 是 1-6, <value> 代表键值的数据。

- 如果 <type> 是 7-8, <value> 代表键值的数据的长度。在你发送完此条命令后, AT 返回 >, 表示 AT 已准备好接收串行数据, 此时你可以输入数据, 当 AT 接收到的数据长度达到 <value> 后, 则立即向命名空间内写入键值对。

说明

- 请先阅读 [非易失性存储 \(NVS\)](#), 了解命名空间、键值对的概念。
- 写入前, 你无需主动擦除命名空间或键值对 (NVS 会根据需要自动擦除键值对)。
- 如果你想修改 mfg_nvs 分区中的某些数据, 请使用 [AT+SYSMFG](#) 命令 (NVS 中的键值对操作)。如果你想修改整个 mfg_nvs 分区, 请使用 [AT+SYSFLASH](#) 命令 (分区操作)。
- 更多 AT+SYSMFG 命令的使用示例, 请参考 [AT+SYSMFG 命令示例](#)。
- 修改配置时, 本命令不会对配置值进行功能性检查。即使你配置了错误的数值, 也会被存储。因此, 你需要自行确保配置数据的正确性。

示例

```
// 向 client_cert 命名空间内的 client_cert.0 键写入新的值 (即: 更新 client_cert_
↳命名空间内的第 0 套客户端证书)
AT+SYSMFG=2,"client_cert","client_cert.0",8,1164

// 等待串口返回 > 后, 写入 1164 字节的证书文件
```

3.1.20 AT+RFPOWER: 查询/设置 RF TX Power

查询命令

功能:

查询 RF TX Power

命令:

```
AT+RFPOWER?
```

响应:

```
+RFPOWER:<wifi_power>,<ble_adv_power>,<ble_scan_power>,<ble_conn_power>
OK
```

设置命令

命令:

```
AT+RFPOWER=<wifi_power>[,<ble_adv_power>,<ble_scan_power>,<ble_conn_power>]
```

响应:

```
OK
```

参数

- **<wifi_power>**: 单位为 0.25 dBm, 比如设定的参数值为 78, 则实际的 RF Power 值为 $78 * 0.25 \text{ dBm} = 19.5 \text{ dBm}$ 。配置后可运行 AT+RFPOWER? 命令确认实际的 RF Power 值。

- ESP32-C6 设备的取值范围为 [40,84]:

设定值	读取值	实际值	实际 dBm
[40,80]	< 设定值 >	< 设定值 >	< 设定值 > * 0.25
[81,84]	< 设定值 >	80	20

- **<ble_adv_power>**: Bluetooth LE 广播的 RF TX Power。取值范围为 [3,15]:
 - 3: -15 dBm
 - 4: -12 dBm
 - 5: -9 dBm
 - 6: -6 dBm
 - 7: -3 dBm
 - 8: 0 dBm
 - 9: 3 dBm
 - 10: 6 dBm
 - 11: 9 dBm
 - 12: 12 dBm
 - 13: 15 dBm
 - 14: 18 dBm
 - 15: 20 dBm
- **<ble_scan_power>**: Bluetooth LE 扫描的 RF TX Power，参数取值同 <ble_adv_power> 参数。
- **<ble_conn_power>**: Bluetooth LE 连接的 RF TX Power，参数取值同 <ble_adv_power> 参数。

说明

- 当 Wi-Fi 关闭或未初始化时，AT+RFPower 命令无法设置/查询 Wi-Fi 的 RF TX Power。当 Bluetooth LE 未初始化时，AT+RFPower 命令无法设置/查询 Bluetooth LE 的 RF TX Power。
- 由于 RF TX Power 分为不同的等级，而每个等级都有与之对应的取值范围，所以通过 esp_wifi_get_max_tx_power 查询到的 wifi_power 的值可能与 esp_wifi_set_max_tx_power 设定的值存在差异，但不会比该值大。
- 建议将 <ble_scan_power> 和 <ble_conn_power> 两个参数值设置为与 <ble_adv_power> 参数相同的值，否则，这两个参数将会被自动设置为与 <ble_adv_power> 相同的值。
- 配置更改不保存到 flash。如果需要保存到 flash，请参考[最大发射功率操作](#)。

3.1.21 AT+RFCAL: RF 全面校准

执行命令

命令:

```
AT+RFCAL
```

响应:

```
OK
```

说明

- ESP32-C6 首次启动时会自动执行 RF 全面校准，之后启动时会自动执行 RF 部分校准。请参考[RF 校准](#)了解更多细节。
- 通常在固件升级、设备环境改变，设备长期未使用等情况后，建议执行 RF 全面校准。

3.1.22 AT+SYSROLLBACK: 回滚到以前的固件

查询命令

功能:

查询当前运行固件和回滚固件的地址和版本。

命令:

```
AT+SYSROLLBACK?
```

响应:

```
+SYSROLLBACK:<running_app_addr>,<"running_app_version">,<rollback_app_addr>,<
↪"rollback_app_version">
OK
```

执行命令

功能:

回滚到以前的固件。

命令:

```
AT+SYSROLLBACK
```

响应:

```
OK
```

参数

- **<running_app_addr>**: 当前运行固件的地址。
- **<"running_app_version">**: 当前运行固件的版本。
- **<rollback_app_addr>**: 回滚固件的地址。
- **<"rollback_app_version">**: 回滚固件的版本。

说明

- 在执行固件回滚时，ESP32-C6 会先验证目标固件的有效性，只有验证通过时才会执行回滚操作，否则将返回错误信息。
- 建议在补丁版本间回滚，不建议跨大版本回滚。跨大版本回滚可能存在一定的兼容性问题，甚至无法运行，如果你坚持要跨大版本回滚，请根据自己的产品自行测试验证功能。
- 回滚功能独立于 OTA 升级流程，可直接跳转至另一 OTA 分区中的固件版本运行。

3.1.23 AT+SYSTIMESTAMP: 查询/设置本地时间戳

查询命令

功能:

查询本地时间戳

命令:


```
AT+SYSTIMESTAMP?
```

响应:

```
+SYSTIMESTAMP:<Unix_timestamp>  
OK
```

设置命令

功能:

设置本地时间戳，当 SNTP 时间更新后，将与之同步更新

命令:

```
AT+SYSTIMESTAMP=<Unix_timestamp>
```

响应:

```
OK
```

参数

- **<Unix-timestamp>**: Unix 时间戳，单位：秒。

示例

```
AT+SYSTIMESTAMP=1565853509 //2019-08-15 15:18:29
```

3.1.24 AT+SYSLOG: 启用或禁用 AT 错误代码提示

查询命令

功能:

查询 AT 错误代码提示是否启用

命令:

```
AT+SYSLOG?
```

响应:

```
+SYSLOG:<status>  
OK
```

设置命令

功能:

启用或禁用 AT 错误代码提示

命令:

```
AT+SYSLOG=<status>
```

响应:

```
OK
```

参数

- **<status>**: 错误代码提示状态
 - 0: 禁用
 - 1: 启用

示例

```
// 启用 AT 错误代码提示
AT+SYSLOG=1

OK
AT+FAKE
ERR CODE:0x01090000

ERROR
```

```
// 禁用 AT 错误代码提示
AT+SYSLOG=0

OK
AT+FAKE
// 不提示 `ERR CODE:0x01090000`

ERROR
```

AT 错误代码是一个 32 位十六进制数值，定义如下：

类型	子类型	扩展
bit 32 ~ bit 24	bit 23 ~ bit 16	bit 15 ~ bit 0

- **category**: 固定值 0x01
- **subcategory**: 错误类型

错误类型	错误代码	说明
ESP_AT_SUB_OK	0x00	OK
ESP_AT_SUB_COMMON_ERROR	0x01	保留
ESP_AT_SUB_NO_TERMINATOR	0x02	未找到结束符（应以“rn”结尾）
ESP_AT_SUB_NO_AT	0x03	未找到起始 AT（输入的可能是 at、At 或 aT）
ESP_AT_SUB_PARA_LENGTH_MISM	0x04	参数长度不匹配
ESP_AT_SUB_PARA_TYPE_MISMAT	0x05	参数类型不匹配
ESP_AT_SUB_PARA_NUM_MISMAT	0x06	参数数量不匹配
ESP_AT_SUB_PARA_INVALID	0x07	无效参数
ESP_AT_SUB_PARA_PARSE_FAIL	0x08	解析参数失败
ESP_AT_SUB_UNSUPPORT_CMD	0x09	不支持该命令
ESP_AT_SUB_CMD_EXEC_FAIL	0x0A	执行命令失败
ESP_AT_SUB_CMD_PROCESSING	0x0B	仍在执行上一条命令
ESP_AT_SUB_CMD_OP_ERROR	0x0C	命令操作类型错误

- **extension:** 错误扩展信息，不同的子类型有不同的扩展信息，详情请见 components/at/include/esp_at.h。

例如，错误代码 ERR_CODE:0x01090000 表示“不支持该命令”。

3.1.25 AT+SLEEPWKCFG: 设置 Light-sleep 唤醒源和唤醒 GPIO

设置命令

命令:

```
AT+SLEEPWKCFG=<wakeup source>,<param1>[,<param2>]
```

响应:

```
OK
```

参数

- **<wakeup source>:** 唤醒源
 - 0: 保留配置，暂不支持
 - 1: 保留配置，暂不支持
 - 2: GPIO 唤醒
- **<param1>:**
 - 当唤醒源为 GPIO 时，该参数表示 GPIO 管脚
 - 保留配置，暂不支持
- **<param2>:**
 - 当唤醒源为 GPIO 时，该参数表示唤醒电平
 - * 0: 低电平
 - * 1: 高电平

示例

```
// GPIO12 置为低电平时唤醒
AT+SLEEPWKCFG=2,12,0
```

说明

- 唤醒管脚的电平应为有效电平，不能处于浮空状态，必须保持高电平或低电平。

3.1.26 AT+SYSSTORE: 设置参数存储模式

查询命令

功能:

查询 AT 参数存储模式

命令:

```
AT+SYSSTORE?
```

响应:

```
+SYSSTORE:<store_mode>
OK
```

设置命令

命令:

```
AT+SYSSTORE=<store_mode>
```

响应:

```
OK
```

参数

- **<store_mode>**: 参数存储模式
 - 0: 命令配置不存入 flash
 - 1: 命令配置存入 flash (默认)

说明

- 该命令只影响设置命令，不影响查询命令，因为查询命令总是从 RAM 中调用。
- 本命令会影响以下命令:

- *AT+SYMSG*
- *AT+CWMODE*
- *AT+CWBANDWIDTH*
- *AT+CIPV6*
- *AT+CWCONFIG*
- *AT+CWJAP*
- *AT+CWSAP*
- *AT+CWRECONCFG*
- *AT+CIPAP*
- *AT+CIPSTA*
- *AT+CIPAPMAC*
- *AT+CIPSTAMAC*

- *AT+CIPDNS*
- *AT+CIPSSLCONF*
- *AT+CIPRECONNINTV*
- *AT+CIPTCPOPT*
- *AT+CWDHCPS*
- *AT+CWDHCP*
- *AT+CWSTAPROTO*
- *AT+CWAPPROTO*
- *AT+CWJEAP*

示例

```
AT+SYSSTORE=0
AT+CWMODE=1 // 不存入 flash
AT+CWJAP="test","1234567890" // 不存入 flash

AT+SYSSTORE=1
AT+CWMODE=3 // 存入 flash
AT+CWJAP="test","1234567890" // 存入 flash
```

3.1.27 AT+SYSREG: 读写寄存器

设置命令

命令:

```
AT+SYSREG=<direct>,<address>[,<write value>]
```

响应:

```
+SYSREG:<read value> // 仅适用于读寄存器时
OK
```

参数

- **<direct>**: 读或写寄存器
 - 0: 读寄存器
 - 1: 写寄存器
- **<address>**: (uint32) 寄存器地址, 详情请参考相关的《技术参考手册》
- **<write value>**: (uint32) 写入值, 仅适用于写寄存器时

说明

- AT 不检查寄存器地址, 因此请确保操作的寄存器地址有效

3.2 Wi-Fi AT 命令集

- [介绍](#)
- *AT+CWINIT*: 初始化/清理 Wi-Fi 驱动程序
- *AT+CWMODE*: 查询/设置 Wi-Fi 模式 (Station/SoftAP/Station+SoftAP)

- [AT+CWBANDWIDTH](#): 查询/设置 Wi-Fi 带宽
- [AT+CWSTATE](#): 查询 Wi-Fi 状态和 Wi-Fi 信息
- [AT+CWCONFIG](#): 查询/设置 Wi-Fi 非活动时间和监听间隔时间
- [AT+CWJAP](#): 连接 AP
- [AT+CWRECONNCFG](#): 查询/设置 Wi-Fi 重连配置
- [AT+CWLAPOPT](#): 设置 [AT+CWLAP](#) 命令扫描结果的属性
- [AT+CWLAP](#): 扫描当前可用的 AP
- [AT+CWQAP](#): 断开与 AP 的连接
- [AT+CWSAP](#): 配置 ESP32-C6 SoftAP 参数
- [AT+CWLIF](#): 查询连接到 ESP32-C6 SoftAP 的 station 信息
- [AT+CWQIF](#): 断开 station 与 ESP32-C6 SoftAP 的连接
- [AT+CWDHCP](#): 启用或禁用 DHCP
- [AT+CWDHCPS](#): 查询/设置 ESP32-C6 SoftAP DHCP 分配的 IPv4 地址范围
- [AT+CWAUTOCONN](#): 上电是否自动连接 AP
- [AT+CWAPPROTO](#): 查询/设置 SoftAP 模式下 Wi-Fi 协议标准
- [AT+CWSTAPROTO](#): 查询/设置 Station 模式下 Wi-Fi 协议标准
- [AT+CIPSTAMAC](#): 查询/设置 ESP32-C6 Station 的 MAC 地址
- [AT+CIPAPMAC](#): 查询/设置 ESP32-C6 SoftAP 的 MAC 地址
- [AT+CIPSTA](#): 查询/设置 ESP32-C6 Station 的 IP 地址
- [AT+CIPAP](#): 查询/设置 ESP32-C6 SoftAP 的 IP 地址
- [AT+CWSTARTSMART](#): 开启 SmartConfig
- [AT+CWSTOPSMART](#): 停止 SmartConfig
- [AT+WPS](#): 设置 WPS 功能
- [AT+CWJEAP](#): 连接 WPA2/WPA3 企业版 AP
- [AT+CWHOSTNAME](#): 查询/设置 ESP32-C6 Station 的主机名称
- [AT+CWCOUNTRY](#): 查询/设置 Wi-Fi 国家代码

3.2.1 介绍

重要: 默认的 AT 固件支持此页面下除 [AT+CWJEAP](#) 之外的所有 AT 命令。如果你需要修改 ESP32-C6 默认支持的命令，请自行编译 [ESP-AT 工程](#)，在第五步配置工程里选择（下面每项是独立的，根据你的需要选择）：

- 启用 EAP 命令 ([AT+CWJEAP](#)): Component config>AT>AT Wi-Fi Enterprise command support
- 禁用 WPS 命令 ([AT+WPS](#)): Component config>AT>AT WPS command support
- 禁用 smartconfig 命令 ([AT+CWSTARTSMART](#)、[AT+CWSTOPSMART](#)): Component config>AT>AT smartconfig command support
- 禁用所有 Wi-Fi 命令（不推荐。一旦禁用，所有 Wi-Fi 以及以上的功能将无法使用，你需要自行实现这些 AT 命令）: Component config>AT>AT wifi command support

3.2.2 AT+CWINIT: 初始化/清理 Wi-Fi 驱动程序

查询命令

功能:

查询 ESP32-C6 设备的 Wi-Fi 初始化状态

命令:

```
AT+CWINIT?
```

响应:

```
+CWINIT:<init>
```

```
OK
```

设置命令

功能:

初始化或清理 ESP32-C6 设备的 Wi-Fi 驱动程序

命令:

```
AT+CWINIT=<init>
```

响应:

```
OK
```

参数

- **<init>:**
 - 0: 清理 Wi-Fi 驱动程序
 - 1: 初始化 Wi-Fi 驱动程序（默认值）

说明

- 本设置不保存到 flash，重启后会恢复为默认值 1。
- 当你 RAM 资源不足时，在不使用 Wi-Fi 的前提下，可以使用此命令清理 Wi-Fi 驱动程序，以释放 RAM 资源。

示例

```
// 清理 Wi-Fi 驱动程序  
AT+CWINIT=0
```

3.2.3 AT+CWMODE: 查询/设置 Wi-Fi 模式 (Station/SoftAP/Station+SoftAP)

查询命令

功能:

查询 ESP32-C6 设备的 Wi-Fi 模式

命令:

```
AT+CWMODE?
```

响应:

```
+CWMODE:<mode>  
OK
```

设置命令

功能:

设置 ESP32-C6 设备的 Wi-Fi 模式

命令:

```
AT+CWMODE=<mode>[, <auto_connect>]
```

响应:

```
OK
```

参数

- **<mode>**: 模式
 - 0: 无 Wi-Fi 模式, 并且关闭 Wi-Fi RF
 - 1: Station 模式
 - 2: SoftAP 模式
 - 3: SoftAP+Station 模式
- **<auto_connect>**: 切换 ESP32-C6 设备的 Wi-Fi 模式时 (例如, 从 SoftAP 或无 Wi-Fi 模式切换为 Station 模式或 SoftAP+Station 模式), 是否启用自动连接 AP 的功能, 默认值: 1。参数缺省时, 使用默认值, 也就是能自动连接。
 - 0: 禁用自动连接 AP 的功能
 - 1: 启用自动连接 AP 的功能, 若之前已经将自动连接 AP 的配置保存到 flash 中, 则 ESP32-C6 设备将自动连接 AP

说明

- 若 [AT+SYSTORE=1](#), 本设置将保存在 NVS 分区
- 如你之前使用过蓝牙功能, 为获得更好的性能, 建议在使用 SoftAP 或 SoftAP+Station 功能前, 先发送以下命令注销已初始化过的功能:
 - [AT+BLEINIT=0](#) (注销 Bluetooth LE)
 - [AT+BLUFI=0](#) (关闭 BluFi)
 如你想了解更多细节, 请阅读 [RF 共存](#) 文档。

示例

```
AT+CWMODE=3
```

3.2.4 AT+CWBANDWIDTH: 查询/设置 Wi-Fi 带宽

查询命令

命令:

```
AT+CWBANDWIDTH?
```

响应:

```
+CWBANDWIDTH:<netif>,<bandwidth_2ghz>,<bandwidth_5ghz>
```

```
OK
```


设置命令

命令:

```
AT+CWBANDWIDTH=<netif>,<bandwidth_2ghz>[,<bandwidth_5ghz>]
```

响应:

```
OK
```

参数

- **<netif>**:
 - 0: Station 接口
 - 1: SoftAP 接口
- **<bandwidth_2ghz>**: 2.4 GHz 带宽
 - 0: 在设置命令中表示保持默认值, 在查询命令中表示当前值无效
 - 1: 20 MHz
 - 2: 40 MHz
- **<bandwidth_5ghz>**: 5 GHz 带宽
 - 0: 在设置命令中表示保持默认值, 在查询命令中表示当前值无效
 - 1: 20 MHz
 - 2: 40 MHz

说明

- 若 **AT+SYSTORE=1**, 本设置将保存在 NVS 分区
- 默认情况下, ESP32-C6 设备的 Station 模式下 Wi-Fi 带宽是 20 MHz, SoftAP 模式下 Wi-Fi 带宽是 40 MHz
- 注意: Wi-Fi 带宽设置受限于硬件能力和 Wi-Fi 协议标准

3.2.5 AT+CWSTATE: 查询 Wi-Fi 状态和 Wi-Fi 信息

查询命令

功能:

查询 ESP32-C6 设备的 Wi-Fi 状态和 Wi-Fi 信息

命令:

```
AT+CWSTATE?
```

响应:

```
+CWSTATE:<state>,<"ssid">
```

```
OK
```

参数

- **<state>**: 当前 Wi-Fi 状态
 - 0: ESP32-C6 station 尚未进行任何 Wi-Fi 连接
 - 1: ESP32-C6 station 已经连接上 AP, 但尚未获取到 IPv4 地址

- 2: ESP32-C6 station 已经连接上 AP，并已经获取到 IPv4 地址
- 3: ESP32-C6 station 正在进行 Wi-Fi 连接或 Wi-Fi 重连
- 4: ESP32-C6 station 处于 Wi-Fi 断开状态
- `<"ssid">`: 目标 AP 的 SSID

说明

- 当 ESP32-C6 station 没有连接上 AP 时，推荐使用此命令查询 Wi-Fi 信息；当 ESP32-C6 station 已连接上 AP 后，推荐使用 [AT+CWJAP](#) 命令查询 Wi-Fi 信息

3.2.6 AT+CWCONFIG: 查询/设置 Wi-Fi 非活动时间和监听间隔时间

查询命令

命令:

```
AT+CWCONFIG?
```

响应:

```
+CWCONFIG:<ap_inactive_time>,<sta_inactive_time>,<listen_interval>
OK
```

设置命令

命令:

```
AT+CWCONFIG=[<ap_inactive_time>][,<sta_inactive_time>][,<listen_interval>]
```

响应:

```
OK
```

参数

- `<ap_inactive_time>`: SoftAP 模式下的非活动时间。单位: 秒, 默认值: 300, 范围: [10,3600]。如果 SoftAP 在非活动时间内没有接收到某个已连接的 Station 的任何数据, SoftAP 将强制断开该 Station 的 Wi-Fi 连接。
- `<sta_inactive_time>`: Station 模式下的非活动时间。单位: 秒, 默认值: 6, 范围: [3,600]。如果 Station 在非活动时间内没有接收到已连接的 SoftAP 的任何信标帧, 将强制断开 ESP32-C6 的 Wi-Fi 连接。
- `<listen_interval>`: 监听 AP beacon 的间隔, 单位为 AP beacon 间隔, 默认值: 3, 范围: [1,100]。和 [AT+CWJAP](#) 中的 `<listen_interval>` 参数相同, 但此配置是全局性的。

说明

- 若 [AT+SYSTORE=1](#), 本设置将保存在 NVS 分区

3.2.7 AT+CWJAP: 连接 AP

查询命令

功能:

查询与 ESP32-C6 Station 连接的 AP 信息

命令:

```
AT+CWJAP?
```

响应:

```
+CWJAP:<"ssid">,<"bssid">,<channel>,<rssi>,<pci_en>,<reconn_interval>,<listen_
↪interval>,<scan_mode>,<pmf>
OK
```

设置命令

功能:

设置 ESP32-C6 Station 需连接的 AP

命令:

```
AT+CWJAP=[<"ssid">],[<"pwd">],[<"bssid">][,<authmode>][,<reconn_interval>][,
↪<listen_interval>][,<scan_mode>][,<jap_timeout>][,<pmf>]
```

响应:

```
WIFI CONNECTED
WIFI GOT IP

OK
[WIFI GOT IPv6 LL]
[WIFI GOT IPv6 GL]
```

或

```
+CWJAP:<error code>
ERROR
```

执行命令

功能:

将 ESP32-C6 station 连接至上次 Wi-Fi 配置中的 AP

命令:

```
AT+CWJAP
```

响应:

```
WIFI CONNECTED
WIFI GOT IP

OK
[WIFI GOT IPv6 LL]
[WIFI GOT IPv6 GL]
```

或

```
+CWJAP:<error code>
ERROR
```

参数

- **<"ssid">**: 目标 AP 的 SSID
 - 如果 SSID 和密码中有 , 、 " 、 \ 等特殊字符, 需转义。
 - AT 支持连接 SSID 为中文的 AP, 但是某些路由器或者热点的中文 SSID 不是 UTF-8 编码格式。你可以先扫描 SSID, 然后使用扫描到的 SSID 进行连接。
- **<"pwd">**: 密码最长 64 字节 ASCII
- **<"bssid">**: 目标 AP 的 MAC 地址, 当多个 AP 有相同的 SSID 时, 该参数不可省略
- **<channel>**: 信道号
- **<rssi>**: 信号强度
- **<authmode>**: Wi-Fi 认证模式阈值。ESP-AT 仅连接认证模式等于或高于此阈值的 AP
 - 0: OPEN (当 <"pwd"> 未设置或为空时) 或 WEP (当 <"pwd"> 长度大于等于 8 字节时)
 - 1: WPA_PSK
 - 2: WPA_PSK
 - 3: WPA2_PSK
 - 4: WPA_WPA2_PSK
 - 5: WPA2_ENTERPRISE
 - 6: WPA3_PSK
 - 7: WPA2_WPA3_PSK
 - 8: WAPI_PSK
 - 9: OWE
 - 10: WPA3_ENT_192
 - 11: WPA3_EXT_PSK
 - 12: WPA3_EXT_PSK_MIXED_MODE
 - 13: DPP
 - 14: WPA3_ENTERPRISE
 - 15: WPA2_WPA3_ENTERPRISE
 - 16: WPA_ENTERPRISE
- **<reconn_interval>**: Wi-Fi 重连间隔, 单位: 秒, 默认值: 1, 最大值: 7200
 - 0: 断开连接后, ESP32-C6 station 不重连 AP
 - [1,7200]: 断开连接后, ESP32-C6 station 每隔指定的时间与 AP 重连
- **<listen_interval>**: 监听 AP beacon 的间隔, 单位为 AP beacon 间隔, 默认值: 3, 范围: [1,100]
- **<scan_mode>**: 扫描模式
 - 0: 快速扫描, 找到目标 AP 后终止扫描, ESP32-C6 station 与第一个扫描到的 AP 连接
 - 1: 全信道扫描, 所有信道都扫描后才终止扫描, ESP32-C6 station 与扫描到的信号最强的 AP 连接
- **<jap_timeout>**: [AT+CWJAP](#) 命令超时的最大值, 单位: 秒, 默认值: 15, 范围: [3,600]
- **<pmf>**: PMF (Protected Management Frames, 受保护的管理帧), 默认值 1
 - 0 表示禁用 PMF
 - bit 0: 具有 PMF 功能, 提示支持 PMF, 如果其他设备具有 PMF 功能, 则 ESP32-C6 设备将优先选择以 PMF 模式连接
 - bit 1: 需要 PMF, 提示需要 PMF, 设备将不会关联不支持 PMF 功能的设备
- **<error code>**: 错误码, 仅供参考
 - 1: 连接超时
 - 2: 密码错误
 - 3: 无法找到目标 AP
 - 4: 连接失败
 - 其它值: 发生未知错误

说明

- 如果 [AT+SYSTORE=1](#), 配置更改将保存到 NVS 分区。

- 使用本命令需要开启 station 模式。
- 当 ESP32-C6 station 已连接上 AP 后，推荐使用此命令查询 Wi-Fi 信息；当 ESP32-C6 station 没有连接上 AP 时，推荐使用 [AT+CWSTATE](#) 命令查询 Wi-Fi 信息。
- 本命令中的 <reconn_interval> 参数与 [AT+CWRECONNCFG](#) 命令中的 <interval_second> 参数相同。如果运行本命令时不设置 <reconn_interval> 参数，Wi-Fi 重连间隔时间将采用默认值 1。
- 如果同时省略 <"ssid"> 和 <"password"> 参数，将使用上一次设置的值。
- 执行命令与设置命令的超时时间相同，默认为 15 秒，可通过参数 <jap_timeout> 设置。
- 不支持通过 [WAPI](#) 鉴权方式连接路由器。
- 想要获取 IPv6 地址，需要先设置 [AT+CIPV6=1](#)。
- 回复 OK 代表 IPv4 网络已经准备就绪，而不代表 IPv6 网络准备就绪。当前 ESP-AT 以 IPv4 网络为主，IPv6 网络为辅。
- WIFI GOT IPv6 LL 代表已经获取到本地链路 IPv6 地址，这个地址是通过 EUI-64 本地计算出来的，不需要路由器参与。由于并行时序，这个打印可能在 OK 之前，也可能在 OK 之后。
- WIFI GOT IPv6 GL 代表已经获取到全局 IPv6 地址，该地址是由 AP 下发的前缀加上内部计算出来的后缀进行组合而来的，需要路由器参与。由于并行时序，这个打印可能在 OK 之前，也可能在 OK 之后；也可能由于 AP 不支持 IPv6 而不打印。

示例

```
// 如果目标 AP 的 SSID 是 "abc", 密码是 "0123456789", 则命令是:
AT+CWJAP="abc","0123456789"

// 如果目标 AP 的 SSID 是 "ab\\,c", 密码是 "0123456789\\", 则命令是:
AT+CWJAP="ab\\\\,c","0123456789\\\\"

// 如果多个 AP 有相同的 SSID "abc", 可通过 BSSID 找到目标 AP:
AT+CWJAP="abc","0123456789","ca:d7:19:d8:a6:44"

// 如果 ESP-AT 要求通过 PMF 连接 AP, 则命令是:
AT+CWJAP="abc","0123456789",,,,,,,,,,3
```

3.2.8 AT+CWRECONNCFG: 查询/设置 Wi-Fi 重连配置

查询命令

功能:

查询 Wi-Fi 重连配置

命令:

```
AT+CWRECONNCFG?
```

响应:

```
+CWRECONNCFG:<interval_second>,<repeat_count>
OK
```

设置命令

功能:

设置 Wi-Fi 重连配置

命令:

```
AT+CWRECONNCFG=<interval_second>,<repeat_count>[,<reconnect_now>]
```

响应:

```
OK
```

参数

- **<interval_second>**: Wi-Fi 重连间隔, 单位: 秒, 默认值: 0, 最大值 7200
 - 0: 断开连接后, ESP32-C6 station 不重连 AP
 - [1,7200]: 断开连接后, ESP32-C6 station 每隔指定的时间与 AP 重连
- **<repeat_count>**: ESP32-C6 设备尝试重连 AP 的次数, 本参数在 <interval_second> 不为 0 时有效, 默认值: 0, 最大值: 1000
 - 0: ESP32-C6 station 始终尝试连接 AP
 - [1,1000]: ESP32-C6 station 按照本参数指定的次数重连 AP
- **<reconnect_now>**: 设置完成后, 是否立即开始重连 AP, 默认值: 0。注意: 仅当 ESP32-C6 的 Wi-Fi 尚未获取到 IP 地址时, 才会重连; 若已获取到 IP 地址, 则保持当前连接状态不变
 - 0: 不立即重连 AP
 - 1: 立即重连 AP

示例

```
// ESP32-C6 station 每隔 1 秒尝试重连 AP, 共尝试 100 次
AT+CWRECONNCFG=1,100

// ESP32-C6 station 在断开连接后不重连 AP
AT+CWRECONNCFG=0,0

// 命令设置完成后, 立即开始重连 AP
AT+CWRECONNCFG=2,50,1
```

说明

- 若 **AT+SYSSTORE=1**, 配置更改将保存在 NVS 分区
- 本命令中的 <interval_second> 参数与 **AT+CWJAP** 中的 [<reconn_interval>] 参数相同
- 该命令适用于被动断开 AP、Wi-Fi 模式切换和开机后 Wi-Fi 自动连接
- 建议在 Wi-Fi 连接成功后再执行此命令

3.2.9 AT+CWLAPOPT: 设置 AT+CWLAP 命令扫描结果的属性

设置命令

命令:

```
AT+CWLAPOPT=<reserved>,<print mask>[,<rssi filter>][,<authmode mask>]
```

响应:

```
OK
```

或者

```
ERROR
```

参数

- **<reserved>**: 保留项
- **<print mask>**: **AT+CWLAP** 的扫描结果是否显示以下参数，默认值: 0x7FF, 若 bit 设为 1, 则显示对应参数, 若设为 0, 则不显示对应参数
 - bit 0: 是否显示 <ecn>
 - bit 1: 是否显示 <" ssid" >
 - bit 2: 是否显示 <rssi>
 - bit 3: 是否显示 <" mac" >
 - bit 4: 是否显示 <channel>
 - bit 5: 是否显示 <freq_offset>
 - bit 6: 是否显示 <freqcal_val>
 - bit 7: 是否显示 <pairwise_cipher>
 - bit 8: 是否显示 <group_cipher>
 - bit 9: 是否显示 <bgn>
 - bit 10: 是否显示 <wps>
- **<rssi filter>**: **AT+CWLAP** 的扫描结果是否按照本参数过滤, 也即, 是否过滤掉信号强度低于 rssi filter 参数值的 AP, 单位: dBm, 默认值: -100, 范围: [-100,40]
- **<authmode mask>**: **AT+CWLAP** 的扫描结果是否显示以下认证方式的 AP, 默认值: 0xFFFF, 如果 bit x 设为 1, 则显示对应认证方式的 AP, 若设为 0, 则不显示
 - bit 0: 是否显示 OPEN 认证方式的 AP
 - bit 1: 是否显示 WEP 认证方式的 AP
 - bit 2: 是否显示 WPA_PSK 认证方式的 AP
 - bit 3: 是否显示 WPA2_PSK 认证方式的 AP
 - bit 4: 是否显示 WPA_WPA2_PSK 认证方式的 AP
 - bit 5: 是否显示 WPA2_ENTERPRISE 认证方式的 AP
 - bit 6: 是否显示 WPA3_PSK 认证方式的 AP
 - bit 7: 是否显示 WPA2_WPA3_PSK 认证方式的 AP
 - bit 8: 是否显示 WAPI_PSK 认证方式的 AP
 - bit 9: 是否显示 OWE 认证方式的 AP
 - bit 10: 是否显示 WPA3_ENT_192 认证方式的 AP
 - bit 11: 是否显示 WPA3_EXT_PSK 认证方式的 AP
 - bit 12: 是否显示 WPA3_EXT_PSK_MIXED_MODE 认证方式的 AP
 - bit 13: 是否显示 DPP 认证方式的 AP
 - bit 14: 是否显示 WPA3_ENTERPRISE 认证方式的 AP
 - bit 15: 是否显示 WPA2_WPA3_ENTERPRISE 认证方式的 AP

示例

```
// 第一个参数缺省
// 第二个参数为 31, 即 0x1F, 表示所有值为 1 的 bit 对应的参数都会显示出来
AT+CWLAPOPT=,31
AT+CWLAP

// 只显示认证方式为 OPEN 的 AP
AT+CWLAPOPT=,31,-100,1
AT+CWLAP
```

3.2.10 AT+CWLAP: 扫描当前可用的 AP

设置命令

功能:

列出符合特定条件的 AP, 如指定 SSID、MAC 地址或信道号

命令:

```
AT+CWLAP=[<"ssid">][,<"mac">][,<channel>][,<scan_type>][,<scan_time_min>][,<scan_time_max>][,<2ghz_channel_bitmap>]
```

执行命令

功能:

列出当前可用的 AP

命令:

```
AT+CWLAP
```

响应:

```
+CWLAP: (<ecn>,<"ssid">,<rssi>,<"mac">,<channel>,<freq_offset>,<freqcal_val>,<pairwise_cipher>,<group_cipher>,<wifi_protocol>,<wps>)
```

```
OK
```

参数

- **<ecn>**: 加密方式
 - 0: OPEN
 - 1: WEP
 - 2: WPA_PSK
 - 3: WPA2_PSK
 - 4: WPA_WPA2_PSK
 - 5: WPA2_ENTERPRISE
 - 6: WPA3_PSK
 - 7: WPA2_WPA3_PSK
 - 8: WAPI_PSK
 - 9: OWE
 - 10: WPA3_ENT_192
 - 11: WPA3_EXT_PSK
 - 12: WPA3_EXT_PSK_MIXED_MODE
 - 13: DPP
 - 14: WPA3_ENTERPRISE
 - 15: WPA2_WPA3_ENTERPRISE
- **<"ssid">**: 字符串参数, AP 的 SSID
- **<rssi>**: 信号强度
- **<"mac">**: 字符串参数, AP 的 MAC 地址
- **<channel>**: 信道号, 指定后仅扫描该信道
- **<scan_type>**: Wi-Fi 扫描类型, 默认值为: 0
 - 0: 主动扫描
 - 1: 被动扫描
- **<scan_time_min>**: 每个信道最短扫描时间, 单位: 毫秒, 范围: [0,1500], 如果扫描类型为被动扫描, 本参数无效
- **<scan_time_max>**: 每个信道最长扫描时间, 单位: 毫秒, 范围: [0,1500], 如果设为 0, 固件采用参数默认值, 主动扫描为 120 ms, 被动扫描为 360 ms
- **<2ghz_channel_bitmap>**: 2.4 GHz 频段信道的位图。在 <channel> 缺省时或为 0 时生效。每一位对应一个信道, 1 表示扫描该信道, 0 表示不扫描该信道。2.4 GHz 频段支持的信道见: [2.4G 信道](#)。例如, 若 bit 1 和 bit 2 设为 1, 则表示扫描信道 1 和信道 2
- **<freq_offset>**: 频偏 (保留项目)
- **<freqcal_val>**: 频率校准值 (保留项目)
- **<pairwise_cipher>**: 成对加密类型

- 0: None
- 1: WEP40
- 2: WEP104
- 3: TKIP
- 4: CCMP
- 5: TKIP and CCMP
- 6: AES-CMAC-128
- 7: 未知
- **<group_cipher>**: 组加密类型, 与 <pairwise_cipher> 参数的枚举值相同
- **<wifi_protocol>**: Wi-Fi 协议标准, 若 bit 设为 1, 则表示使能对应模式, 若设为 0, 则表示禁用对应模式
 - bit 0: 是否使能 802.11b 模式
 - bit 1: 是否使能 802.11g 模式
 - bit 2: 是否使能 802.11n 模式
 - bit 3: [802.11 LR 乐鑫专利协议标准](#)
 - bit 4: 802.11ax 协议标准
 - bit 5: 802.11a 协议标准
 - bit 6: 802.11ac 协议标准
- **<wps>**: wps flag
 - 0: 不支持 WPS
 - 1: 支持 WPS

说明

- 当 SSID 中包含 , 、 " 或 \ 字符时, 需对上述字符进行转义后再配置。
- 若无法扫描到中文 SSID 的 AP, 请确保 SSID 的字符编码与路由器配置一致 (通常为 UTF-8)。
- 当扫描结果异常时, 建议使用抓包工具分析 Probe Request 报文中的 SSID 字段, 检查 <"ssid"> 参数是否与目标 AP 的 SSID 完全匹配。

示例

```
AT+CWLAP="Wi-Fi", "ca:d7:19:d8:a6:44", 6, 0, 400, 1000
```

```
// 寻找指定 SSID 的 AP
AT+CWLAP="Wi-Fi"
```

3.2.11 AT+CWQAP: 断开与 AP 的连接

执行命令

命令:

```
AT+CWQAP
```

响应:

```
OK
```

3.2.12 AT+CWSAP: 配置 ESP32-C6 SoftAP 参数

查询命令

功能:

查询 ESP32-C6 SoftAP 的配置参数

命令:

```
AT+CWSAP?
```

响应:

```
+CWSAP:<"ssid">,<"pwd">,<channel>,<ecn>,<max conn>,<ssid hidden>
OK
```

设置命令**功能:**

设置 ESP32-C6 SoftAP 的配置参数

命令:

```
AT+CWSAP=<"ssid">,<"pwd">,<chl>,<ecn>[,<max conn>][,<ssid hidden>]
```

响应:

```
OK
```

参数

- <"ssid">: 字符串参数, 接入点名称
- <"pwd">: 字符串参数, 密码, 范围: 8 ~ 64 字节 ASCII
- <channel>: 信道号
- <ecn>: 加密方式, 不支持 WEP
 - 0: OPEN
 - 2: WPA_PSK
 - 3: WPA2_PSK
 - 4: WPA_WPA2_PSK
- <max conn>: 允许连入 ESP32-C6 SoftAP 的最多 station 数目, 取值范围: 参考 [max_connection 描述](#)。
- <ssid hidden>:
 - 0: 广播 SSID (默认)
 - 1: 不广播 SSID

说明

- 本命令只有当 `AT+CWMODE=2` 或者 `AT+CWMODE=3` 时才有效。
- 若 `AT+SYSTORE=1`, 配置更改将保存在 NVS 分区。
- 默认 SSID 因设备而异, 因为它由设备的 MAC 地址组成。你可以使用 `AT+CWSAP?` 查询默认的 SSID。

示例

```
AT+CWSAP="ESP","1234567890",5,3
```

3.2.13 AT+CWLIF: 查询连接到 ESP32-C6 SoftAP 的 station 信息**执行命令****命令:**

```
AT+CWLIF
```

响应:

```
+CWLIF:<ip addr>,<mac>  
OK
```

参数

- **<ip addr>**: 连接到 ESP32-C6 SoftAP 的 station 的 IP 地址
- **<mac>**: 连接到 ESP32-C6 SoftAP 的 station 的 MAC 地址

说明

- 本命令无法查询静态 IP，仅支持在 ESP32-C6 SoftAP 和连入的 station DHCP 均使能的情况下有效

3.2.14 AT+CWQIF: 断开 station 与 ESP32-C6 SoftAP 的连接

执行命令

功能:

断开所有连入 ESP32-C6 SoftAP 的 station

命令:

```
AT+CWQIF
```

响应:

```
OK
```

设置命令

功能:

断开某个连入 ESP32-C6 SoftAP 的 station

命令:

```
AT+CWQIF=<"mac">
```

响应:

```
OK
```

参数

- **<" mac" >**: 需断开连接的 station 的 MAC 地址

3.2.15 AT+CWDHCP: 启用或禁用 DHCP

查询命令

命令:

```
AT+CWDHCP?
```

响应:

```
+CWDHCP:<state>  
OK
```

设置命令

功能:

启用或禁用 DHCP

命令:

```
AT+CWDHCP=<operate>,<mode>
```

响应:

```
OK
```

参数

- **<operate>**:
 - 0: 禁用
 - 1: 启用
- **<mode>**:
 - bit 0: Station 的 DHCP
 - bit 1: SoftAP 的 DHCP
- **<state>**: DHCP 的状态
 - bit 0:
 - * 0: 禁用 Station 的 DHCP
 - * 1: 启用 Station 的 DHCP
 - bit 1:
 - * 0: 禁用 SoftAP 的 DHCP
 - * 1: 启用 SoftAP 的 DHCP
 - bit 2:
 - * 0: 禁用 Ethernet 的 DHCP
 - * 1: 启用 Ethernet 的 DHCP

说明

- 若 *AT+SYSTORE=1*，配置更改将保存到 NVS 分区
- 本设置命令与设置静态 IPv4 地址的命令会相互影响，如 *AT+CIPSTA* 和 *AT+CIPAP*
 - 若启用 DHCP，则静态 IPv4 地址会被禁用
 - 若启用静态 IPv4，则 DHCP 会被禁用
 - 最后一次配置会覆盖上一次配置

示例

```
// 启用 Station DHCP，如果原 DHCP mode 为 2，则现 DHCP mode 为 3
AT+CWDHCP=1,1

// 禁用 SoftAP DHCP，如果原 DHCP mode 为 3，则现 DHCP mode 为 1
AT+CWDHCP=0,2
```

3.2.16 AT+CWDHCPS：查询/设置 ESP32-C6 SoftAP DHCP 分配的 IPv4 地址范围

查询命令

命令：

```
AT+CWDHCPS?
```

响应：

```
+CWDHCPS:<lease time>,<start IP>,<end IP>
OK
```

设置命令

功能：

设置 ESP32-C6 SoftAP DHCP 服务器分配的 IPv4 地址范围

命令：

```
AT+CWDHCPS=<enable>,<lease time>,<"start IP">,<"end IP">
```

响应：

```
OK
```

参数

- **<enable>**:
 - 1: 设置 DHCP server 信息，后续参数必须填写
 - 0: 清除 DHCP server 信息，恢复默认值，后续参数无需填写
- **<lease time>**: 租约时间，单位：分钟，取值范围：[1,2880]
- **<" start IP" >**: ESP32-C6 SoftAP DHCP 服务器 IPv4 地址池的起始 IP
- **<" end IP" >**: ESP32-C6 SoftAP DHCP 服务器 IPv4 地址池的结束 IP

说明

- 若 **AT+SYSTORE=1**，配置更改将保存到 NVS 分区
- 本命令必须在 ESP32-C6 SoftAP 模式使能，且开启 DHCP server 的情况下使用
- 设置的 IPv4 地址范围必须与 ESP32-C6 SoftAP 在同一网段

示例

```
AT+CWDHCPS=1,3,"192.168.4.10","192.168.4.15"

AT+CWDHCPS=0 // 清除设置，恢复默认值
```

3.2.17 AT+CWAUTOCONN: 查询/设置上电是否自动连接 AP

查询命令

命令:

```
AT+CWAUTOCONN?
```

响应:

```
+CWAUTOCONN:<enable>  
OK
```

设置命令

命令:

```
AT+CWAUTOCONN=<enable>
```

响应:

```
OK
```

参数

- **<enable>**:
 - 1: 上电自动连接 AP (默认)
 - 0: 上电不自动连接 AP

说明

- 本设置保存到 NVS 区域。
- 对功耗敏感的场景, 建议关闭上电自动连接 AP, 或在 [AT+CWRECONNCFG](#) 中将 <repeat_count> 设置为较小值, 以避免上电后持续重连导致的额外耗电。

示例

```
AT+CWAUTOCONN=1
```

3.2.18 AT+CWAPPROTO: 查询/设置 SoftAP 模式下 Wi-Fi 协议标准

查询命令

命令:

```
AT+CWAPPROTO?
```

响应:

```
+CWAPPROTO:<protocol_2g>  
OK
```

设置命令

命令:

```
AT+CWAPPROTO=<protocol_2g>
```

响应:

```
OK
```

参数

- **<protocol_2g>**: 2.4 GHz 频段下的 Wi-Fi 协议标准
 - bit 0: 802.11b 协议标准
 - bit 1: 802.11g 协议标准
 - bit 2: 802.11n 协议标准
 - bit 3: [802.11 LR 乐鑫专利协议标准](#)
 - bit 4: 802.11ax 协议标准

说明

- 若 [AT+SYSSTORE=1](#)，配置更改将保存到 NVS 分区
- 当前，ESP32-C6 设备支持的 Wi-Fi 协议标准见: [Wi-Fi 协议模式](#)
- 默认情况下，ESP32-C6 设备的 SoftAP 协议标准是 802.11bgn 协议
- 注意：Wi-Fi 协议标准受限于硬件能力和 Wi-Fi 带宽设置。例如：40 MHz 带宽模式下，仅可配置 802.11n（2.4 GHz）

3.2.19 AT+CWSTAPROTO: 查询/设置 Station 模式下 Wi-Fi 协议标准

查询命令

命令:

```
AT+CWSTAPROTO?
```

响应:

```
+CWSTAPROTO:<protocol_2g>  
OK
```

设置命令

命令:

```
AT+CWSTAPROTO=<protocol_2g>
```

响应:

```
OK
```

参数

- **<protocol_2g>**: 2.4 GHz 频段下的 Wi-Fi 协议标准
 - bit 0: 802.11b 协议标准
 - bit 1: 802.11g 协议标准
 - bit 2: 802.11n 协议标准
 - bit 3: [802.11 LR 乐鑫专利协议标准](#)
 - bit 4: 802.11ax 协议标准

说明

- 若 **AT+SYSTORE=1**，配置更改将保存到 NVS 分区
- 当前，ESP32-C6 设备支持的 Wi-Fi 协议标准见：[Wi-Fi 协议模式](#)
- 默认情况下，ESP32-C6 设备的 Station 协议标准是 802.11bgnax 协议
- 注意：Wi-Fi 协议标准受限于硬件能力和 Wi-Fi 带宽设置。例如：40 MHz 带宽模式下，仅可配置 802.11n（2.4 GHz）

3.2.20 AT+CIPSTAMAC：查询/设置 ESP32-C6 Station 的 MAC 地址

查询命令

功能：

查询 ESP32-C6 Station 的 MAC 地址

命令：

```
AT+CIPSTAMAC?
```

响应：

```
+CIPSTAMAC:<"mac">  
OK
```

设置命令

功能：

设置 ESP32-C6 Station 的 MAC 地址

命令：

```
AT+CIPSTAMAC=<"mac">
```

响应：

```
OK
```

参数

- **<" mac" >**: 字符串参数，表示 ESP32-C6 Station 的 MAC 地址

说明

- 若 `AT+SYSTORE=1`，配置更改将保存到 NVS 分区
- ESP32-C6 Station 的 MAC 地址与 ESP32-C6 SoftAP 不同，不要为二者设置同样的 MAC 地址
- MAC 地址的 bit 0 不能为 1，例如，MAC 地址可以是 “1a:…” ，但不可以是 “15:…”
- FF:FF:FF:FF:FF:FF 和 00:00:00:00:00:00 是无效地址，不能设置

示例

```
AT+CIPSTAMAC="1a:fe:35:98:d3:7b"
```

3.2.21 AT+CIPAPMAC：查询/设置 ESP32-C6 SoftAP 的 MAC 地址

查询命令

功能：

查询 ESP32-C6 SoftAP 的 MAC 地址

命令：

```
AT+CIPAPMAC?
```

响应：

```
+CIPAPMAC:<"mac">  
OK
```

设置命令

功能：

设置 ESP32-C6 SoftAP 的 MAC 地址

命令：

```
AT+CIPAPMAC=<"mac">
```

响应：

```
OK
```

参数

- `<" mac" >`：字符串参数，表示 ESP32-C6 SoftAP 的 MAC 地址

说明

- 若 `AT+SYSTORE=1`，配置更改将保存到 NVS 分区
- ESP32-C6 SoftAP 的 MAC 地址与 ESP32-C6 Station 不同，不要为二者设置同样的 MAC 地址
- MAC 地址的 bit 0 不能为 1，例如，MAC 地址可以是 “18:…” ，但不可以是 “15:…”
- FF:FF:FF:FF:FF:FF 和 00:00:00:00:00:00 是无效地址，不能设置

示例

```
AT+CIPAPMAC="18:fe:35:98:d3:7b"
```

3.2.22 AT+CIPSTA: 查询/设置 ESP32-C6 Station 的 IP 地址

查询命令

功能:

查询 ESP32-C6 Station 的 IP 地址

命令:

```
AT+CIPSTA?
```

响应:

```
+CIPSTA:ip:<"ip">
+CIPSTA:gateway:<"gateway">
+CIPSTA:netmask:<"netmask">
+CIPSTA:ip6ll:<"ipv6 addr">
+CIPSTA:ip6gl:<"ipv6 addr">

OK
```

设置命令

功能:

设置 ESP32-C6 Station 的 IPv4 地址

命令:

```
AT+CIPSTA=<"ip">[,<"gateway">,<"netmask">]
```

响应:

```
OK
```

参数

- <"ip">: 字符串参数, 表示 ESP32-C6 station 的 IPv4 地址
- <"gateway">: 网关
- <"netmask">: 子网掩码
- <"ipv6 addr">: ESP32-C6 station 的 IPv6 地址

说明

- 使用查询命令时, 只有当 ESP32-C6 station 连入 AP 或者配置过静态 IP 地址后, 才能查询到它的 IP 地址
- 若 *AT+SYSSSTORE=1*, 配置更改将保存到 NVS 分区
- 本设置命令与设置 DHCP 的命令相互影响, 如 *AT+CWDHCP*
 - 若启用静态 IPv4 地址, 则禁用 DHCP
 - 若启用 DHCP, 则禁用静态 IPv4 地址
 - 最后一次配置会覆盖上一次配置
- 使用静态 IP 时, AT 不提供 IP 冲突检测功能, 如有需要请自行实现

示例

```
AT+CIPSTA="192.168.6.100","192.168.6.1","255.255.255.0"
```

3.2.23 AT+CIPAP: 查询/设置 ESP32-C6 SoftAP 的 IP 地址

查询命令

功能:

查询 ESP32-C6 SoftAP 的 IP 地址

命令:

```
AT+CIPAP?
```

响应:

```
+CIPAP:ip:<"ip">
+CIPAP:gateway:<"gateway">
+CIPAP:netmask:<"netmask">
+CIPAP:ip6ll:<"ipv6 addr">

OK
```

设置命令

功能:

设置 ESP32-C6 SoftAP 的 IPv4 地址

命令:

```
AT+CIPAP=<"ip">[,<"gateway">,<"netmask">]
```

响应:

```
OK
```

参数

- <"ip">: 字符串参数, 表示 ESP32-C6 SoftAP 的 IPv4 地址
- <"gateway">: 网关
- <"netmask">: 子网掩码
- <"ipv6 addr">: ESP32-C6 SoftAP 的 IPv6 地址

说明

- 本设置命令仅适用于 IPv4 网络, 不适用于 IPv6 网络
- 若 *AT+SYSTORE=1*, 配置更改将保存到 NVS 分区
- 本设置命令与设置 DHCP 的命令相互影响, 如 *AT+CWDHCP*
 - 若启用静态 IPv4 地址, 则禁用 DHCP
 - 若启用 DHCP, 则禁用静态 IPv4 地址
 - 最后一次配置会覆盖上一次配置

示例

```
AT+CIPAP="192.168.5.1","192.168.5.1","255.255.255.0"
```

3.2.24 AT+CWSTARTSMART: 开启 SmartConfig

执行命令

功能:

开启 ESP-TOUCH+AirKiss 兼容模式

命令:

```
AT+CWSTARTSMART
```

设置命令

功能:

开启某指定类型的 SmartConfig

命令:

```
AT+CWSTARTSMART=<type>[,<auth floor>][,<"esptouch v2 key">]
```

响应:

```
OK
```

参数

- **<type>**: 类型
 - 1: ESP-TOUCH
 - 2: AirKiss
 - 3: ESP-TOUCH+AirKiss
 - 4: ESP-TOUCH v2
- **<auth floor>**: Wi-Fi 认证模式阈值, ESP-AT 不会连接到 authmode 低于此阈值的 AP
 - 0: OPEN (默认)
 - 1: WEP
 - 2: WPA_PSK
 - 3: WPA2_PSK
 - 4: WPA_WPA2_PSK
 - 5: WPA2_ENTERPRISE
 - 6: WPA3_PSK
 - 7: WPA2_WPA3_PSK
 - 8: WAPI_PSK
 - 9: OWE
 - 10: WPA3_ENT_192
 - 11: WPA3_EXT_PSK
 - 12: WPA3_EXT_PSK_MIXED_MODE
 - 13: DPP
 - 14: WPA3_ENTERPRISE
 - 15: WPA2_WPA3_ENTERPRISE
- **<" esptouch v2 key" >**: ESP-TOUCH v2 的解密密钥, 用于解密 Wi-Fi 密码和自定义数据。长度应为 16 字节。

说明

- 更多有关 SmartConfig 的信息，请参考 [ESP-TOUCH 使用指南](#)；
- SmartConfig 仅支持在 ESP32-C6 Station 模式下调用；
- 消息 Smart get Wi-Fi info 表示 SmartConfig 成功获取到 AP 信息，之后 ESP32-C6 尝试连接 AP；
- 消息 +SCRD:<length>,<rvd data> 表示 ESP-Touch v2 成功获取到自定义数据；
- 消息 Smartconfig connected Wi-Fi 表示成功连接到 AP；
- 因为 ESP32-C6 设备需要将 SmartConfig 配网结果同步给手机端，所以建议在消息 Smartconfig connected Wi-Fi 输出后延迟超过 6 秒再调用 [AT+CWSTOPSMART](#)；
- 可调用 [AT+CWSTOPSMART](#) 停止 SmartConfig，然后再执行其他命令。注意，在 SmartConfig 过程中请勿执行其他命令。

示例

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

3.2.25 AT+CWSTOPSMART: 停止 SmartConfig

执行命令

命令:

```
AT+CWSTOPSMART
```

响应:

```
OK
```

说明

- 无论 SmartConfig 成功与否，都请在执行其他命令之前调用 [AT+CWSTOPSMART](#) 释放 SmartConfig 占用的内存

示例

```
AT+CWMODE=1
AT+CWSTARTSMART
AT+CWSTOPSMART
```

3.2.26 AT+WPS: 设置 WPS 功能

设置命令

命令:

```
AT+WPS=<enable>[,<auth floor>]
```

响应:

```
OK
```

参数

- **<enable>**:
 - 1: 开启 PBC 类型的 WPS
 - 0: 关闭 PBC 类型的 WPS
- **<auth floor>**: Wi-Fi 认证模式阈值, ESP-AT 不会连接到 authmode 低于此阈值的 AP
 - 0: OPEN (默认)
 - 1: WEP
 - 2: WPA_PSK
 - 3: WPA2_PSK
 - 4: WPA_WPA2_PSK
 - 5: WPA2_ENTERPRISE
 - 6: WPA3_PSK
 - 7: WPA2_WPA3_PSK
 - 8: WAPI_PSK
 - 9: OWE
 - 10: WPA3_ENT_192
 - 11: WPA3_EXT_PSK
 - 12: WPA3_EXT_PSK_MIXED_MODE
 - 13: DPP
 - 14: WPA3_ENTERPRISE
 - 15: WPA2_WPA3_ENTERPRISE

说明

- WPS 功能必须在 ESP32-C6 Station 使能的情况下调用
- WPS 不支持 WEP 加密方式

示例

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

3.2.27 AT+CWJEAP: 连接 WPA2/WPA3 企业版 AP

查询命令

功能:

查询 ESP32-C6 station 连入的企业版 AP 的配置信息

命令:

```
AT+CWJEAP?
```

响应:

```
+CWJEAP:<"ssid">,<method>,<"identity">,<"username">,<"password">,<security>
OK
```

设置命令

功能:

连接到目标企业版 AP

命令:

```
AT+CWJEAP=<"ssid">,<method>,<"identity">,<"username">,<"password">,<security>[,  
↪<jeap_timeout>][,<rssi>][,<"bssid">][,<channel>][,<scan_mode>][,<auth_mode>]
```

响应:

```
OK
```

或

```
+CWJEAP:Timeout  
ERROR
```

参数

- **<"ssid">**: 企业版 AP 的 SSID
 - 如果 SSID 或密码中包含 ,、"、\\ 等特殊字符, 需转义
- **<method>**: WPA2/WPA3 企业版认证方式
 - 0: EAP-TLS
 - 1: EAP-PEAP
 - 2: EAP-TTLS
- **<"identity">**: 阶段 1 的身份, 字符串限制为 1~32
- **<"username">**: 阶段 2 的用户名, 范围: 1~32 字节, EAP-PEAP、EAP-TTLS 两种认证方式需设置本参数, EAP-TLS 方式无需设置本参数
- **<"password">**: 阶段 2 的密码, 范围: 1~32 字节, EAP-PEAP、EAP-TTLS 两种认证方式需设置本参数, EAP-TLS 方式无需设置本参数
- **<security>**:
 - bit 0: 提供客户端证书供企业版服务器端校验
 - bit 1: 客户端载入 CA 证书来校验企业版服务器端的证书
- **<jeap_timeout>**: [AT+CWJEAP](#) 命令的最大超时时间, 单位: 秒, 默认值: 15, 范围: [3,600]
- **<rssi>**: 连接时的 RSSI 阈值, Wi-Fi 仅连接 RSSI 不低于该值的 AP。单位: dBm, 范围: [-128,127], 默认值: -127
- **<"bssid">**: 指定目标 AP 的 MAC 地址, 实现精确连接, 避免连接同名但不同 MAC 的 AP。格式为 "xx:xx:xx:xx:xx:xx"
- **<channel>**: 指定扫描起始的信道。从该信道开始, 寻找指定 SSID 的 AP
- **<scan_mode>**: 扫描模式, 默认值为 0
 - 0: 快速扫描, 扫描到对应的 AP 后立即连接, 加快连接速度
 - 1: 全信道扫描, 选择信号最强的 AP 连接
- **<auth_mode>**: 可接受的最低企业版安全等级 (门槛), 默认值为 0。
 - 0: WPA2 企业版
 - 1: WPA3 企业版
 - 2: WPA3 企业版 192-bit (NSA Suite-B)

示例

```
// 连接至 WPA2 企业版 AP (EAP-TLS  
↪认证方式), 设置身份, 验证服务器证书, 加载客户端证书 (校验服务器证书时请先执行  
↪AT+SYSTIMESTAMP 设置时间)  
AT+CWJEAP="dlink11111",0,"example@espressif.com",,,3  
  
// 连接至 WPA2 企业版 AP (EAP-PEAP  
↪认证方式), 设置身份、用户名、密码, 不验证服务器证书, 不加载客户端证书  
AT+CWJEAP="dlink11111",1,"example@espressif.com","espressif","test11",0  
  
// 连接至 WPA3 企业版 AP (EAP-PEAP 方式), 校验服务器证书 (请先执行  
↪AT+SYSTIMESTAMP 设置时间)  
AT+CWJEAP="dlink11111",1,"example@espressif.com","espressif","test11",2,,,,,1
```

```
// 连接至 WPA3 企业版 192-bit (NSA Suite-B) AP (EAP-TLS 方式) 。
// 需使用满足 Suite-B 要求的证书 (EC P-384, 或 RSA 密钥长度 >= 3072, 且使用 SHA-
→ 384 签名), 并先执行 AT+SYSTIMESTAMP 设置时间。
AT+CWJEAP="dlink11111",0,"example@espressif.com",,,,3,,,,,2
```

错误代码:

WPA2/WPA3 企业版错误码以 ERR CODE:0x<%08x> 格式打印:

AT_EAP_MALLOC_FAILED	0x8001
AT_EAP_GET_NVS_CONFIG_FAILED	0x8002
AT_EAP_CONN_FAILED	0x8003
AT_EAP_SET_WIFI_CONFIG_FAILED	0x8004
AT_EAP_SET_IDENTITY_FAILED	0x8005
AT_EAP_SET_USERNAME_FAILED	0x8006
AT_EAP_SET_PASSWORD_FAILED	0x8007
AT_EAP_GET_CA_LEN_FAILED	0x8008
AT_EAP_READ_CA_FAILED	0x8009
AT_EAP_SET_CA_FAILED	0x800A
AT_EAP_GET_CERT_LEN_FAILED	0x800B
AT_EAP_READ_CERT_FAILED	0x800C
AT_EAP_GET_KEY_LEN_FAILED	0x800D
AT_EAP_READ_KEY_FAILED	0x800E
AT_EAP_SET_CERT_KEY_FAILED	0x800F
AT_EAP_ENABLE_FAILED	0x8010
AT_EAP_ALREADY_CONNECTED	0x8011
AT_EAP_GET_SSID_FAILED	0x8012
AT_EAP_SSID_NULL	0x8013
AT_EAP_SSID_LEN_ERROR	0x8014
AT_EAP_GET_METHOD_FAILED	0x8015
AT_EAP_CONN_TIMEOUT	0x8016
AT_EAP_GET_IDENTITY_FAILED	0x8017
AT_EAP_IDENTITY_LEN_ERROR	0x8018
AT_EAP_GET_USERNAME_FAILED	0x8019
AT_EAP_USERNAME_LEN_ERROR	0x801A
AT_EAP_GET_PASSWORD_FAILED	0x801B
AT_EAP_PASSWORD_LEN_ERROR	0x801C
AT_EAP_GET_SECURITY_FAILED	0x801D
AT_EAP_SECURITY_ERROR	0x801E
AT_EAP_METHOD_SECURITY_UNMATCHED	0x801F
AT_EAP_PARAMETER_COUNTS_ERROR	0x8020
AT_EAP_GET_WIFI_MODE_ERROR	0x8021
AT_EAP_WIFI_MODE_NOT_STA	0x8022
AT_EAP_SET_CONFIG_FAILED	0x8023
AT_EAP_METHOD_ERROR	0x8024
AT_EAP_GET_CMD_TIMEOUT_FAILED	0x8025
AT_EAP_CMD_TIMEOUT_ERROR	0x8026
AT_EAP_GET_RSSI_FAILED	0x8027
AT_EAP_RSSI_ERROR	0x8028
AT_EAP_GET_BSSID_FAILED	0x8029
AT_EAP_BSSID_ERROR	0x802A
AT_EAP_GET_CHANNEL_FAILED	0x802B
AT_EAP_CHANNEL_ERROR	0x802C
AT_EAP_GET_SCAN_MODE_FAILED	0x802D
AT_EAP_SCAN_MODE_ERROR	0x802E
AT_EAP_GET_AUTH_MODE_FAILED	0x802F

续下页

表 1 - 接上页

AT_EAP_MALLOC_FAILED	0x8001
AT_EAP_AUTH_MODE_ERROR	0x8030
AT_EAP_SET_TTLS_PHASE2_FAILED	0x8031
AT_EAP_SET_WPA3_ENT_192BIT_FAILED	0x8032

说明

- 若 **AT+SYSTORE=1**，配置更改将保存到 NVS 分区。
- 使用本命令需开启 Station 模式。
- 使用 TLS 认证方式需使能客户端证书。
- 如果你想使用自己的证书，运行时请使用 **AT+SYSMFG** 命令更新 Wi-Fi Enterprise 客户端证书（具体步骤请参考 **AT+SYSMFG** 命令示例，证书配置方法与 SSL 证书相同）。如果你想预烧录自己的证书，请参考 **如何更新 PKI 配置**。
- 如果 <security> 的 bit 1 置位（即取值为 2 或 3），为了校验服务器的证书有效期，请在发送 **AT+CWJEAP** 命令前确保 ESP32-C6 已获取到当前时间（你可以发送 **AT+SYSTIMESTAMP=<unix_timestamp>** 命令来配置当前时间，发送 **AT+SYSTIMESTAMP?** 命令查询当前时间）。
- <auth_mode> 仅为最低门槛，无法强制锁定某一安全等级。最终的安全等级由 AP 的广播能力协商决定，因此设备实际连接的等级可能高于 <auth_mode>。
- WPA3 企业版 192-bit (<auth_mode> = 2) 要求 CA、客户端证书、服务器证书都必须满足 NSA Suite-B 要求（EC P-384，或 RSA 密钥长度 ≥ 3072，且使用 SHA-384 签名），否则将连接失败。

3.2.28 AT+CWHOSTNAME: 查询/设置 ESP32-C6 Station 的主机名称

查询命令

功能:

查询 ESP32-C6 Station 的主机名称

命令:

```
AT+CWHOSTNAME?
```

响应:

```
+CWHOSTNAME:<hostname>
```

```
OK
```

设置命令

功能:

设置 ESP32-C6 Station 的主机名称

命令:

```
AT+CWHOSTNAME=<"hostname">
```

响应:

```
OK
```

若没开启 Station 模式，则返回：

ERROR

参数

- **<"hostname">**: ESP32-C6 Station 的主机名称，最大长度：32 字节

说明

- 配置更改不保存到 flash

示例

AT+CWMODE=3
AT+CWHOSTNAME="my_test"

3.2.29 AT+CWCOUNTRY: 查询/设置 Wi-Fi 国家代码

查询命令

功能:

查询 Wi-Fi 国家代码

命令:

AT+CWCOUNTRY?

响应:

+CWCOUNTRY:<country_policy>,<"country_code">,<start_channel>,<total_channel_count>
OK

设置命令

功能:

设置 Wi-Fi 国家代码

命令:

AT+CWCOUNTRY=<country_policy>,<"country_code">,<start_channel>,<total_channel_
↪count>

响应:

OK

参数

- **<country_policy>**:
 - 0: 将国家代码改为 ESP32-C6 设备连入的 AP 的国家代码。

- 1: 不改变国家代码，始终保持本命令设置的国家代码。
- **<" country_code">**: 国家代码，最大长度：3 个字符，各国国家代码请参考 [ISO 3166-1 alpha-2](#) 标准。
- **<start_channel>**: 起始信道，范围：[1,14]。
- **<total_channel_count>**: 信道总个数。

说明

- 详细说明请参考：[Wi-Fi 国家/地区代码](#)。
- 配置更改不保存到 flash。如果需要保存到 flash，请参考[国家代码操作](#)。

示例

```
AT+CWMODE=3
AT+CWCOUNTRY=1,"CN",1,13
```

3.3 TCP/IP AT 命令

- [介绍](#)
- **AT+CIPV6**: 启用或禁用 IPv6 网络 (IPv6)
- **AT+CIPSTATE**: 查询 TCP/UDP/SSL 连接信息
- **AT+CIPDOMAIN**: 域名解析
- **AT+CIPSTART**: 建立 TCP 连接、UDP 传输或 SSL 连接
- **AT+CIPSTARTEX**: 建立自动分配 ID 的 TCP 连接、UDP 传输或 SSL 连接
- **[仅适用数据模式] +++**: 退出数据模式
- **AT+SAVETRANSLINK**: 设置 Network 开机透传模式信息
- **AT+CIPSEND**: 在普通传输模式或 Network 透传模式下发送数据
- **AT+CIPSENDL**: 在普通传输模式下并行发送长数据
- **AT+CIPSENDLCFG**: 设置 AT+CIPSENDL 命令的属性
- **AT+CIPSENDEX**: 在普通传输模式下采用扩展的方式发送数据
- **AT+CIPCLOSE**: 关闭 TCP/UDP/SSL 连接
- **AT+CIPCONNPERISIST**: 查询、设置 TCP/SSL 连接持久化参数
- **AT+CIFSR**: 查询本地 IP 地址和 MAC 地址
- **AT+CIPMUX**: 启用或禁用多连接模式
- **AT+CIPSERVER**: 建立/关闭 TCP 或 SSL 服务器
- **AT+CIPSERVERMAXCONN**: 查询/设置服务器允许建立的最大连接数
- **AT+CIPMODE**: 查询/设置传输模式
- **AT+CIPSTO**: 查询/设置本地 TCP 服务器超时时间
- **AT+CIPSNTPCFG**: 查询/设置时区和 SNTP 服务器
- **AT+CIPSNTPTIME**: 查询 SNTP 时间
- **AT+CIPSNTPTINTV**: 查询/设置 SNTP 时间同步的间隔
- **AT+CIPFWVER**: 查询服务器已有的 AT 固件版本
- **AT+CIUPDATE**: 通过 Network 升级固件
- **AT+CIPDINFO**: 设置 +IPD 消息详情
- **AT+CIPSSLCCONF**: 查询/设置 SSL 客户端配置
- **AT+CIPSSLCCIPHER**: 查询/设置 SSL 客户端的密码套件 (cipher suite)
- **AT+CIPSSLCCN**: 查询/设置 SSL 客户端的公用名 (common name)
- **AT+CIPSSLCSNI**: 查询/设置 SSL 客户端的 SNI
- **AT+CIPSSLCALPN**: 查询/设置 SSL 客户端 ALPN
- **AT+CIPSSLCPK**: 查询/设置 SSL 客户端的 PSK (字符串格式)
- **AT+CIPSSLCPKHES**: 查询/设置 SSL 客户端的 PSK (十六进制格式)
- **AT+CIPRECONNINTV**: 查询/设置 Network 透传模式下的 TCP/UDP/SSL 重连间隔
- **AT+CIPRECVTYPE**: 查询/设置套接字接收模式

- **AT+CIPRECVDATA**: 获取被动接收模式下的套接字数据
- **AT+CIPRECLEN**: 查询被动接收模式下套接字数据的长度
- **AT+PING**: ping 对端主机
- **AT+CIPDNS**: 查询/设置 DNS 服务器信息
- **AT+MDNS**: 设置 mDNS 功能
- **AT+CIPTCPOPT**: 查询/设置套接字选项

3.3.1 介绍

重要: 默认的 AT 固件支持此页面下的所有 AT 命令。如果你需要修改 ESP32-C6 默认支持的命令，请自行编译 **ESP-AT** 工程，在第五步配置工程里选择（下面每项是独立的，根据你的需要选择）：

- 禁用 OTA 命令 (**AT+CIUPDATE**、**AT+CIPFWVER**): Component config>AT>AT OTA command support
- 禁用 PING 命令 (**AT+PING**): Component config>AT>AT ping command support
- 禁用 mDNS 命令 (**AT+MDNS**): Component config>AT>AT MDNS command support
- 禁用 TCP/IP 命令（不推荐。一旦禁用，所有 TCP/IP 功能将无法使用，你需要自行实现这些 AT 命令）: Component config>AT>AT net command support

3.3.2 AT+CIPV6: 启用或禁用 IPv6 网络 (IPv6)

查询命令

功能:

查询 IPv6 网络是否使能

命令:

```
AT+CIPV6?
```

响应:

```
+CIPV6:<enable>
```

```
OK
```

设置命令

功能:

启用或禁用 IPv6 网络

命令:

```
AT+CIPV6=<enable>
```

响应:

```
OK
```

参数

- **<enable>**: IPv6 网络使能状态。默认值: 0
 - 0: 禁用 IPv6 网络
 - 1: 启用 IPv6 网络

说明

- 若 `AT+SYSSSTORE=1`，配置更改将保存到 NVS 分区
- 在使用基于 IPv6 网络的上层应用前，需要先启用 IPv6 网络。（例如：基于 IPv6 网络使用 TCP/UDP/SSL/PING/DNS，也称为 TCP6/UDP6/SSL6/PING6/DNS6 或 TCPv6/UDPv6/SSLv6/PINGv6/DNSv6）

3.3.3 AT+CIPSTATE: 查询 TCP/UDP/SSL 连接信息

查询命令

命令:

```
AT+CIPSTATE?
```

响应:

当有连接时，AT 返回:

```
+CIPSTATE:<link ID>,<"type">,<"remote IP">,<remote port>,<local port>,<tetype>
OK
```

当没有连接时，AT 返回:

```
OK
```

参数

- **<link ID>**: 网络连接 ID (0 ~ 4)，用于多连接的情况
- **<" type">**: 字符串参数，表示传输类型: "TCP"、"UDP"、"SSL"、"TCPv6"、"UDPv6" 或 "SSLv6"
- **<" remote IP">**: 字符串参数，表示远端 IPv4 地址或 IPv6 地址
- **<remote port>**: 远端端口值
- **<local port>**: ESP32-C6 本地端口值
- **<tetype>**:
 - 0: ESP32-C6 设备作为客户端
 - 1: ESP32-C6 设备作为服务器

3.3.4 AT+CIPDOMAIN: 域名解析

设置命令

命令:

```
AT+CIPDOMAIN=<"domain name">[,<ip network>][,<timeout>][,<show_all_ip>]
```

响应:

```
+CIPDOMAIN:<"IP address">
OK
```

参数

- **<" domain name">**: 待解析的域名
- **<ip network>**: 首选 IP 网络。默认值: 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
- **<" IP address">**: 解析后的 IPv4 地址或 IPv6 地址。
- **<timeout>**: 命令超时。单位: 毫秒。默认值: 0。范围: [0,60000]。设置为 0 时, 命令的超时依赖于网络和 lwIP 协议栈; 设置为非 0 时, 命令会在指定超时内返回, 但会多消耗约 5 KB 的堆空间。
- **<show_all_ip>**: 是否显示所有解析到的 IP 地址。默认值: 0。
 - 0: 只显示首个解析到的 IP 地址
 - 1: 显示所有解析到的 IP 地址

说明

- 如果你想解析域名为多个 IP 地址, 请增加 python build.py menuconfig > Component config > LWIP > DNS > Maximum number of IP addresses per host 的值, 然后重新编译 ESP-AT 工程。最终返回的格式如下:

```
+CIPDOMAIN:<"IP address">
+CIPDOMAIN:<"IP address">
...
OK
```

示例

```
AT+CWMODE=1 // 设置 station 模式
AT+CWJAP="SSID","password" // 连接网络
AT+CIPDOMAIN="iot.espressif.cn" // 域名解析

// 域名解析, 只解析为 IPv4 地址
AT+CIPDOMAIN="iot.espressif.cn",2

// 域名解析, 只解析为 IPv6 地址
AT+CIPDOMAIN="ipv6.test-ipv6.com",3

// 域名解析, 首选解析为 IPv4 地址
AT+CIPDOMAIN="ds.test-ipv6.com",1
```

3.3.5 AT+CIPSTART: 建立 TCP 连接、UDP 传输或 SSL 连接

- [建立 TCP 连接](#)
- [建立 UDP 传输](#)
- [建立 SSL 连接](#)

建立 TCP 连接

设置命令 命令:

```
// 单连接 (AT+CIPMUX=0):
AT+CIPSTART=<"type">,<"remote host">,<remote port>[,<keep_alive>][,<"local IP">][,
↪<timeout>]

// 多连接 (AT+CIPMUX=1):
AT+CIPSTART=<link ID>,<"type">,<"remote host">,<remote port>[,<keep_alive>][,<
↪"local IP">][,<timeout>]
```

响应:

单连接模式下, 返回:

```
CONNECT
OK
```

多连接模式下, 返回:

```
<link ID>,CONNECT
OK
```

参数

- **<link ID>**: 网络连接 ID (0 ~ 4), 用于多连接的情况。该参数范围取决于 menuconfig 中的两个配置项。一个是 AT 组件中的配置项 AT_SOCKET_MAX_CONN_NUM, 默认值为 5。另一个是 LWIP 组件中的配置项 LWIP_MAX_SOCKETS, 默认值为 10。要修改该参数的范围, 你需要修改配置项 AT_SOCKET_MAX_CONN_NUM 的值并确保该值不大于 LWIP_MAX_SOCKETS 的值。(请参考[编译 ESP-AT 工程](#)获取更多信息)。
- **<" type">**: 字符串参数, 表示网络连接类型, "TCP" 或 "TCPv6"。默认值: "TCP"。
- **<" remote host">**: 字符串参数, 表示远端 IPv4 地址、IPv6 地址, 或域名。最长为 64 字节。如果你需要使用域名且域名长度超过 64 字节, 请使用 [AT+CIPDOMAIN](#) 命令获取域名对应的 IP 地址, 然后使用 IP 地址建立连接。
- **<remote port>**: 远端端口值。
- **<keep_alive>**: 配置套接字的 SO_KEEPALIVE 选项 (参考: [SO_KEEPALIVE 介绍](#)), 单位: 秒。
 - 范围: [0,7200]。
 - * 0: 禁用 keep-alive 功能 (默认)。
 - * 1 ~ 7200: 开启 keep-alive 功能。TCP_KEEPIDLE 值为 <keep_alive>, TCP_KEEPINTVL 值为 1, TCP_KEEPCNT 值为 3。
 - 本命令中的 <keep_alive> 参数与 [AT+CIPTCP](#) 命令中的 <keep_alive> 参数相同, 最终值由后设置的命令决定。如果运行本命令时不设置 <keep_alive> 参数, 则默认使用上次配置的值。
- **<" local IP">**: 本地的 IPv4 地址或 IPv6 地址, 用于绑定连接。使用多个网络接口或多个 IP 地址时, 此参数非常有用。默认为禁用。如需使用请先自行设置。可设置为空。
- **<timeout>**: 命令超时。单位: 毫秒。默认值: 0。范围: [0,60000]。设置为 0 时, 命令的超时依赖于网络和 lwIP 协议栈; 设置为非 0 时, 命令会在指定超时内返回, 但会多消耗约 5 KB 的堆空间。

说明

- 如果你想基于 IPv6 网络建立一个 TCP 连接, 请执行以下操作:
 - 确保 AP 支持 IPv6
 - 设置 [AT+CIPV6=1](#)
 - 通过 [AT+CWJAP](#) 命令获取到一个 IPv6 地址
 - (可选) 通过 [AT+CIPSTA?](#) 命令检查 ESP32-C6 是否获取到 IPv6 地址
- 若需查看建立 TCP 连接失败原因, 请先运行 [AT+SYSLOG=1](#) 启用日志后重试本命令。失败时, AT 会返回更详细的错误码 +ERRNO:<error_code>, 以便定位问题。

示例

```
AT+CIPSTART="TCP","iot.espressif.cn",8000
AT+CIPSTART="TCP","192.168.101.110",1000
AT+CIPSTART="TCP","192.168.101.110",2500,60
AT+CIPSTART="TCP","192.168.101.110",1000,, "192.168.101.100"

// 连接 GitHub 的 TCP 服务器, 设置 5 秒超时
AT+CIPSTART="TCP","www.github.com",80,,,5000
```

(续下页)

(接上页)

```

AT+CIPSTART="TCPv6","test-ipv6.com",80
AT+CIPSTART="TCPv6","fe80::860d:8eff:fe9d:cd90",1000,, "fe80::411c:1fdb:22a6:4d24"

// esp-at 已通过 AT+CWJAP 获取到 IPv6 全局地址
AT+CIPSTART="TCPv6","2404:6800:4005:80b::2004",80,,
↪ "240e:3a1:2070:11c0:32ae:a4ff:fe80:65ac"

```

建立 UDP 传输

设置命令 命令：

```

// 单连接：(AT+CIPMUX=0)
AT+CIPSTART=<"type">,<"remote host">,<remote port>[,<local port>,<mode>,<"local IP
↪ ">][,<timeout>]

// 多连接：(AT+CIPMUX=1)
AT+CIPSTART=<link ID>,<"type">,<"remote host">,<remote port>[,<local port>,<mode>,<
↪ "local IP">][,<timeout>]

```

响应：

单连接模式下，返回：

```

CONNECT
OK

```

多连接模式下，返回：

```

<link ID>,<CONNECT>
OK

```

参数

- **<link ID>**：网络连接 ID (0 ~ 4)，用于多连接的情况。
- **<" type">**：字符串参数，表示网络连接类型，“UDP”或“UDPv6”。默认值：“TCP”。
- **<" remote host">**：字符串参数，表示远端 IPv4 地址、IPv6 地址，或域名。最长为 64 字节。如果你需要使用域名且域名长度超过 64 字节，请使用 *AT+CIPDOMAIN* 命令获取域名对应的 IP 地址，然后使用 IP 地址建立连接。
- **<remote port>**：远端端口值。
- **<local port>**：ESP32-C6 设备的 UDP 端口值。
- **<mode>**：在 UDP 透传下，本参数的值必须设为 0。
 - 0: 接收到 UDP 数据后，不改变对端 UDP 地址信息（默认）
 - 1: 仅第一次接收到与初始设置不同的对端 UDP 数据时，改变对端 UDP 地址信息为发送数据设备的 IP 地址和端口
 - 2: 每次接收到 UDP 数据时，都改变对端 UDP 地址信息为发送数据的设备的 IP 地址和端口
- **<" local IP">**：本地的 IPv4 地址或 IPv6 地址，用于绑定连接。使用多个网络接口或多个 IP 地址时，此参数非常有用。默认为禁用。如需使用请先自行设置。可设置为空。
- **<timeout>**：命令超时。单位：毫秒。默认值：0。范围：[0,60000]。设置为 0 时，命令的超时依赖于网络和 lwIP 协议栈；设置为非 0 时，命令会在指定超时内返回，但会多消耗约 5 KB 的堆空间。

说明

- 如果 UDP 连接中的远端 IP 地址是 IPv4 组播地址 (224.0.0.0 ~ 239.255.255.255)，ESP32-C6 设备将发送和接收 UDPv4 组播。

- 如果 UDP 连接中的远端 IP 地址是 IPv4 广播地址 (255.255.255.255)，ESP32-C6 设备将发送和接收 UDPv4 广播。需要注意的是，AT 不支持自动重复广播，如需定期广播，MCU 需要定时发送数据包。
- 如果 UDP 连接中的远端 IP 地址是 IPv6 组播地址 (FF00:0:0:0:0:0:0 ~ FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF)，ESP32-C6 设备将基于 IPv6 网络，发送和接收 UDP 组播。
- 使用参数 <mode> 前，需先设置参数 <local port>。
- 如果你想基于 IPv6 网络建立一个 UDP 传输，请执行以下操作：
 - 确保 AP 支持 IPv6
 - 设置 `AT+CIPV6=1`
 - 通过 `AT+CWJAP` 命令获取到一个 IPv6 地址
 - (可选) 通过 `AT+CIPSTA?` 命令检查 ESP32-C6 是否获取到 IPv6 地址
- 如果想接收长度大于 1460 字节的 UDP 包，请自行编译 `ESP-AT` 工程，在第五步配置工程里选择：Component config>LWIP>Enable reassembly incoming fragmented IP4 packets。
- 若需查看建立 UDP 传输失败原因，请先运行 `AT+SYSLOG=1` 启用日志后重试本命令。失败时，AT 会返回更详细的错误码 +ERRNO:<error_code>，以便定位问题。

示例

```
// UDPv4 单播
AT+CIPSTART="UDP","192.168.101.110",1000,1002,2
AT+CIPSTART="UDP","192.168.101.110",1000,,,"192.168.101.100"

// 建立和 pool.ntp.org 的 UDP 传输，设置 5 秒超时
AT+CIPSTART="UDP","pool.ntp.org",123,,,,5000

// 基于 IPv6 网络的 UDP 单播
AT+CIPSTART="UDPv6","fe80::32ae:a4ff:fe80:65ac",1000,,,"fe80::5512:f37f:bb03:5d9b"

// 基于 IPv6 网络的 UDP 多播
AT+CIPSTART="UDPv6","FF02::FC",1000,1002,0
```

建立 SSL 连接

设置命令 命令：

```
// 单连接：(AT+CIPMUX=0)
AT+CIPSTART=<"type">,<"remote host">,<remote port>[,<keep_alive>,<"local IP">][,
↪<timeout>]

// 多连接：(AT+CIPMUX=1)
AT+CIPSTART=<link ID>,<"type">,<"remote host">,<remote port>[,<keep_alive>,<"local_
↪IP">][,<timeout>]
```

响应：

单连接模式下，返回：

```
CONNECT
OK
```

多连接模式下，返回：

```
<link ID>,CONNECT
OK
```

参数

- **<link ID>**: 网络连接 ID (0 ~ 4), 用于多连接的情况。
- **<" type">**: 字符串参数, 表示网络连接类型, "SSL" 或 "SSLv6"。默认值: "TCP"。
- **<" remote host">**: 字符串参数, 表示远端 IPv4 地址、IPv6 地址, 或域名。最长为 64 字节。如果你需要使用域名且域名长度超过 64 字节, 请使用 [AT+CIPDOMAIN](#) 命令获取域名对应的 IP 地址, 然后使用 IP 地址建立连接。
- **<remote port>**: 远端端口值。
- **<keep_alive>**: 配置套接字的 SO_KEEPALIVE 选项 (参考: [SO_KEEPALIVE 介绍](#)), 单位: 秒。
 - 范围: [0,7200]。
 - * 0: 禁用 keep-alive 功能 (默认)。
 - * 1 ~ 7200: 开启 keep-alive 功能。TCP_KEEPIDLE 值为 <keep_alive>, TCP_KEEPINTVL 值为 1, TCP_KEEPCNT 值为 3。
 - 本命令中的 <keep_alive> 参数与 [AT+CIPTCPOPT](#) 命令中的 <keep_alive> 参数相同, 最终值由后设置的命令决定。如果运行本命令时不设置 <keep_alive> 参数, 则默认使用上次配置的值。
- **<" local IP">**: 本地的 IPv4 地址或 IPv6 地址, 用于绑定连接。使用多个网络接口或多个 IP 地址时, 此参数非常有用。默认为禁用。如需使用请先自行设置。可设置为空。
- **<timeout>**: 命令超时。单位: 毫秒。默认值: 0。范围: [0,60000]。设置为 0 时, 命令的超时依赖于网络和 lwIP 协议栈; 设置为非 0 时, 命令会在指定超时内返回, 但会多消耗约 5 KB 的堆空间。

说明

- SSL 连接数量取决于可用内存和最大连接数量。
- SSL 连接需占用大量内存, 内存不足会导致系统重启。
- 如果 AT+CIPSTART 命令是基于 SSL 连接, 且每个数据包的超时时间为 10 秒, 则总超时时间会变得更长, 具体取决于握手数据包的个数。
- 有一些 SSL 服务器需要 SSL 客户端在连接时要求拓展字段支持 SNI, 请在 SSL 连接前先使用 [AT+CIPSSLCSCI](#) 命令配置 SNI。通常情况下, SNI 的值为服务器的域名。
- 如果需要调整 TLS 协议版本, 请重新编译 ESP-AT 工程。通过 python build.py menuconfig > Component config > mbedTLS 路径, 禁用不需要的 TLS 版本或启用默认未支持的版本。
 - 例如, 启用 TLS 1.3 协议:
 - python build.py menuconfig > Component config > mbedTLS > Using dynamic TX/RX buffer > 取消勾选
 - python build.py menuconfig > Component config > mbedTLS > mbedTLS v3.x related > Keep peer certificate after handshake completion > 勾选
 - python build.py menuconfig > Component config > mbedTLS > mbedTLS v3.x related > Support TLS 1.3 protocol > 勾选
- 如果你想基于 IPv6 网络建立一个 SSL 连接, 请执行以下操作:
 - 确保 AP 支持 IPv6
 - 设置 [AT+CIPV6=1](#)
 - 通过 [AT+CWJAP](#) 命令获取到一个 IPv6 地址
 - (可选) 通过 [AT+CIPSTA?](#) 命令检查 ESP32-C6 是否获取到 IPv6 地址
- 若需查看建立 SSL 连接失败原因, 请先运行 [AT+SYSLOG=1](#) 启用日志后重试本命令。失败时, AT 会返回更详细的错误码 +ERRNO:<error_code>, 以便定位问题。

示例

```
AT+CIPSTART="SSL","iot.espressif.cn",8443
AT+CIPSTART="SSL","192.168.101.110",1000,, "192.168.101.100"

// 连接微软必应的 SSL 服务器, 设置 5 秒超时
AT+CIPSTART="SSL","www.bing.com",443,,,5000

// esp-at 已通过 AT+CWJAP 获取到 IPv6 全局地址
AT+CIPSTART="SSLv6","240e:3a1:2070:11c0:6972:6f96:9147:d66d",1000,,
↪ "240e:3a1:2070:11c0:55ce:4e19:9649:b75"
```

3.3.6 AT+CIPSTARTEX: 建立自动分配 ID 的 TCP 连接、UDP 传输或 SSL 连接

本命令与 *AT+CIPSTART* 相似，不同点在于：在多连接的情况下 (*AT+CIPMUX=1*) 无需手动分配 ID，系统会自动为新建的连接分配 ID。

3.3.7 [仅适用数据模式] +++: 退出数据模式

特殊执行命令

功能:

退出数据模式，进入命令模式

Command:

```
// 仅适用数据模式
+++
```

说明

- 此特殊执行命令包含有三个相同的 + 字符（即 ASCII 码：0x2b），同时命令结尾没有 CR-LF 字符
- 确保第一个 + 字符前至少有 20 ms 时间间隔内没有其他输入，第三个 + 字符后至少有 20 ms 时间间隔内没有其他输入，三个 + 字符之间至多有 20 ms 时间间隔内没有其他输入。否则，+ 字符会被当做普通数据发送出去
- 本条特殊执行命令没有命令回复
- 请至少间隔 1 秒再发下一条 AT 命令

3.3.8 AT+CIPSEND: 在普通传输模式或 Network 透传模式下发送数据

设置命令

功能:

普通传输模式下，指定长度发送数据。如果你要发送的数据长度大于 8192 字节，请使用 *AT+CIPSENDL* 命令发送。

命令:

```
// 单连接: (AT+CIPMUX=0)
AT+CIPSEND=<length>

// 多连接: (AT+CIPMUX=1)
AT+CIPSEND=<link ID>,<length>

// UDP 传输可指定对端主机和端口
AT+CIPSEND=[<link ID>,<length>[,<"remote host">,<remote port>]
```

响应:

```
OK

>
```

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

如果未建立连接或数据传输时连接被断开，返回：

```
ERROR
```

如果所有数据被成功发往协议栈（不代表数据已经发送到对端），返回：

```
SEND OK
```

执行命令

功能：

进入 Network [透传模式](#)

命令：

```
AT+CIPSEND
```

响应：

```
OK
>
```

或

```
ERROR
```

进入 Network [透传模式](#)，ESP32-C6 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，数据会立即被分为每 2920 字节一组的块进行发送，否则会等待 20 毫秒或等待收到的数据大于等于 2920 字节再发送数据（你可以通过 [AT+TRANSINTVL](#) 命令配置此间隔）。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 秒再发下一条 AT 命令。

本命令必须在开启 [透传模式](#) 以及单连接下使用。若为 UDP 透传，[AT+CIPSTART](#) 命令的参数 `<mode>` 必须设置为 0。

参数

- **<link ID>**：网络连接 ID (0 ~ 4)，用于多连接的情况
- **<length>**：数据长度，最大值：8192 字节
- **<" remote host">**：UDP 传输可以指定对端主机：IPv4 地址、IPv6 地址，或域名
- **<remote port>**：UDP 传输可以指定对端端口

说明

- 你可以使用 [AT+CIPTCPPOPT](#) 命令来为每个 TCP 连接配置套接字选项。例如：设置 `<so_sndtimeo>` 为 5000，则 TCP 发送操作会在 5 秒内返回结果，无论成功还是失败。这可以节省 MCU 等待 AT 命令回复的时间。

3.3.9 AT+CIPSENDL：在普通传输模式下并行发送长数据

设置命令

功能：

[普通传输模式](#) 下，指定长度，并行发送数据（AT 命令端口接收数据和 AT 往对端发送数据是并行的）。你可以使用 [AT+CIPSENDLCFG](#) 命令配置本条命令。如果你要发送的数据长度小于 8192 字节，你也可以使用 [AT+CIPSEND](#) 命令发送。

命令：

```
// 单连接：(AT+CIPMUX=0)
AT+CIPSENDL=<length>

// 多连接：(AT+CIPMUX=1)
AT+CIPSENDL=<link ID>,<length>

// UDP 传输可指定对端主机和端口
AT+CIPSENDL=[<link ID>,<length>[,<"remote host">,<remote port>]
```

响应:

```
OK

>
```

上述响应表示 AT 进入**数据模式** 并且已准备好接收 AT 命令端口的数据，此时你可以输入数据，一旦 AT 命令端口接收到数据，数据就会被发往底层协议，数据传输开始。

如果传输已开始，系统会根据**AT+CIPSENDLCFG** 配置上报消息：

```
+CIPSENDL:<had sent len>,<port recv len>
```

如果传输被+++ 命令取消，系统返回：

```
SEND CANCELLED
```

如果所有数据没有被完全发出去，系统最终返回：

```
SEND FAIL
```

如果所有数据被成功发往协议栈（不代表数据已经发送到对端），系统最终返回：

```
SEND OK
```

当连接断开时，你可以发送+++ 命令取消传输，同时 ESP32-C6 设备会从**数据模式** 退出。否则，AT 命令端口会一直接收数据，直到收到指定的 <length> 长度数据后，才会退出**数据模式**。

参数

- **<link ID>**：网络连接 ID (0 ~ 4)，用于多连接的情况
- **<length>**：数据长度，最大值： $2^{31} - 1$ 字节
- **<"remote host">**：UDP 传输可以指定对端主机：IPv4 地址、IPv6 地址，或域名
- **<remote port>**：UDP 传输可以指定对端端口
- **<had sent len>**：成功发到底层协议栈的数据长度
- **<port recv len>**：AT 命令端口收到的数据总长度

说明

- 建议你使用 UART 流控。否则，如果 UART 接收速度大于网络发送速度时，将会导致数据丢失。
- 你可以使用**AT+CIPTCPPOPT** 命令来为每个 TCP 连接配置套接字选项。例如：设置 <so_sndtimeo> 为 5000，则 TCP 发送操作会在 5 秒内返回结果，无论成功还是失败。这可以节省 MCU 等待 AT 命令回复的时间。

3.3.10 AT+CIPSENDLCFG：设置 AT+CIPSENDL 命令的属性**查询命令****功能:**

查询 [AT+CIPSENDL](#) 命令的配置

命令:

```
AT+CIPSENDLCFG?
```

响应:

```
+CIPSENDLCFG:<report size>,<transmit size>
OK
```

设置命令

功能:

设置 [AT+CIPSENDL](#) 命令的配置

命令:

```
AT+CIPSENDLCFG=<report size>[,<transmit size>]
```

响应:

```
OK
```

参数

- **<report size>**: [AT+CIPSENDL](#) 命令中的上报块大小。默认值: 1024。范围: [100,2²⁰]。例如: 设置 <report size> 值为 100, 则 [AT+CIPSENDL](#) 命令回复里的 <had sent len> 上报序列为 (100, 200, 300, 400, …)。最后的 <had sent len> 上报值总是等于实际传输的数据长度。
- **<transmit size>**: [AT+CIPSENDL](#) 命令中的传输块大小, 它指定了数据发往协议栈的数据块大小。默认值: 2920。范围: [100,2920]。如果收到的数据长度大于等于 <transmit size>, 数据会立即被分为每 <transmit size> 字节一组的块进行发送, 否则会等待 20 毫秒或等待收到的数据大于等于 <transmit size> 字节再发送数据 (你可以通过 [AT+TRANSINTVL](#) 命令配置此间隔)。

说明

- 对于吞吐量小但对实时性要求高的设备, 推荐你设置较小的 <transmit size>。也推荐你通过 [AT+CIPTCPOPT](#) 命令设置 TCP_NODELAY 属性。
- 对于吞吐量大的设备, 推荐你设置较大的 <transmit size>。也推荐你阅读[如何提高 ESP-AT 吞吐性能](#)。

3.3.11 AT+CIPSENDEX: 在普通传输模式下采用扩展的方式发送数据

设置命令

功能:

[普通传输模式](#) 下, 指定长度发送数据, 或者使用字符串 \0 (0x5c, 0x30 ASCII) 触发数据发送

命令:

```
// 单连接: (AT+CIPMUX=0)
AT+CIPSENDEX=<length>

// 多连接: (AT+CIPMUX=1)
AT+CIPSENDEX=<link ID>,<length>
```

(续下页)

(接上页)

```
// UDP 传输可指定对端 IP 地址和端口：
AT+CIPSENDEX=[<link ID>,<length>[,<"remote host">,<remote port>]
```

响应：

```
OK

>
```

上述响应表示 AT 已准备好接收串行数据，此时你可以输入指定长度的数据，当 AT 接收到的数据长度达到 <length> 后或数据中出现 \0 字符时，数据传输开始。

如果未建立连接或数据传输时连接被断开，返回：

```
ERROR
```

如果所有数据被成功发往协议栈（不代表数据已经发送到对端），返回：

```
SEND OK
```

参数

- **<link ID>**：网络连接 ID (0 ~ 4)，用于多连接的情况
- **<length>**：数据长度，最大值：8192 字节
- **<"remote host">**：UDP 传输可以指定对端主机：IPv4 地址、IPv6 地址，或域名
- **<remote port>**：UDP 传输可以指定对端端口

说明

- 当数据长度满足要求时，或数据中出现 \0 字符时 (0x5c, 0x30 ASCII)，数据传输开始，系统返回普通命令模式，等待下一条 AT 命令。
- 如果数据中包含 \<any>，则会去掉反斜杠，只使用 <any> 符号。
- 如果需要发送 \0，请转义为 \\0。
- 你可以使用 **AT+CIPTCPOPT** 命令来为每个 TCP 连接配置套接字选项。例如：设置 <so_sndtimeo> 为 5000，则 TCP 发送操作会在 5 秒内返回结果，无论成功还是失败。这可以节省 MCU 等待 AT 命令回复的时间。

3.3.12 AT+CIPCLOSE：关闭 TCP/UDP/SSL 连接**设置命令****功能：**

关闭多连接模式下的 TCP/UDP/SSL 连接

命令：

```
AT+CIPCLOSE=<link ID>
```

响应：

```
<link ID>,CLOSED

OK
```

执行命令

功能:

关闭单连接模式下的 TCP/UDP/SSL 连接

```
AT+CIPCLOSE
```

响应:

```
CLOSED
```

```
OK
```

参数

- **<link ID>**: 需关闭的网络连接 ID, 如果设为 5, 则表示关闭所有连接

3.3.13 AT+CIPCONNPERSIST: 查询、设置 TCP/SSL 连接持久化参数

查询命令

功能:

查询 TCP/SSL 连接的持久化属性

命令:

```
AT+CIPCONNPERSIST?
```

响应:

```
+CIPCONNPERSIST:<persist_link>[...][,<persist_link>]
```

```
OK
```

设置命令

功能:

设置 TCP/SSL 连接的持久化属性

命令:

```
// 单连接: (AT+CIPMUX=0)
AT+CIPCONNPERSIST=<persist_link>

// 多连接: (AT+CIPMUX=1)
AT+CIPCONNPERSIST=<link ID>,<persist_link>
```

响应:

```
OK
```


参数

- **<link ID>**: 需设置的网络连接 ID。
- **<persist_link>**: 是否启用连接保持。关闭（默认）时，ESP32-C6 在 netif 停止、断开或丢失 IP 后会立即主动关闭该 netif 上的 TCP/SSL 连接；启用时，AT 不会主动断开，由协议栈自行处理连接状态。
 - 0: 禁用连接保持（默认）。
 - 1: 启用连接保持。

3.3.14 AT+CIFSR: 查询本地 IP 地址和 MAC 地址

执行命令

命令:

```
AT+CIFSR
```

响应:

```
+CIFSR:APIP,<"APIP">
+CIFSR:APIP6LL,<"APIP6LL">
+CIFSR:APIP6GL,<"APIP6GL">
+CIFSR:APMAC,<"APMAC">
+CIFSR:STAIP,<"STAIP">
+CIFSR:STAIP6LL,<"STAIP6LL">
+CIFSR:STAIP6GL,<"STAIP6GL">
+CIFSR:STAMAC,<"STAMAC">
+CIFSR:ETHIP,<"ETHIP">
+CIFSR:ETHIP6LL,<"ETHIP6LL">
+CIFSR:ETHIP6GL,<"ETHIP6GL">
+CIFSR:ETHMAC,<"ETHMAC">
```

```
OK
```

参数

- **<" APIP" >**: ESP32-C6 Wi-Fi SoftAP 的 IPv4 地址
- **<" APIP6LL" >**: ESP32-C6 Wi-Fi SoftAP 的 IPv6 本地链路地址
- **<" APIP6GL" >**: ESP32-C6 Wi-Fi SoftAP 的 IPv6 全局地址
- **<" APMAC" >**: ESP32-C6 Wi-Fi SoftAP 的 MAC 地址
- **<" STAIP" >**: ESP32-C6 Wi-Fi station 的 IPv4 地址
- **<" STAIP6LL" >**: ESP32-C6 Wi-Fi station 的 IPv6 本地链路地址
- **<" STAIP6GL" >**: ESP32-C6 Wi-Fi station 的 IPv6 全局地址
- **<" STAMAC" >**: ESP32-C6 Wi-Fi station 的 MAC 地址
- **<" ETHIP" >**: ESP32-C6 ethernet 的 IPv4 地址
- **<" ETHIP6LL" >**: ESP32-C6 ethernet 的 IPv6 本地链路地址
- **<" ETHIP6GL" >**: ESP32-C6 ethernet 的 IPv6 全局地址
- **<" ETHMAC" >**: ESP32-C6 ethernet 的 MAC 地址

说明

- 只有当 ESP32-C6 设备获取到有效接口信息后，才能查询到它的 IP 地址和 MAC 地址

3.3.15 AT+CIPMUX: 启用或禁用多连接模式

查询命令

功能:

查询连接模式

命令:

```
AT+CIPMUX?
```

响应:

```
+CIPMUX:<mode>  
OK
```

设置命令

功能:

设置连接模式

命令:

```
AT+CIPMUX=<mode>
```

响应:

```
OK
```

参数

- **<mode>**: 连接模式, 默认值: 0
 - 0: 单连接
 - 1: 多连接

说明

- 只有当所有连接都断开时才可更改连接模式
- 只有普通传输模式 ([AT+CIPMODE=0](#)), 才能设置为多连接
- 如果建立了 TCP/SSL 服务器, 想切换为单连接, 必须关闭服务器 ([AT+CIPSERVER=0](#))
- 多连接并行传输大量数据时, 可能会出现内存峰值不足的情况, 请你根据产品的实际情况进行测试, 并实现 MCU 对内存不足的处理

示例

```
AT+CIPMUX=1
```

3.3.16 AT+CIPSERVER: 建立/关闭 TCP 或 SSL 服务器

查询命令

功能:

查询 TCP/SSL 服务器状态

命令:

```
AT+CIPSERVER?
```

响应:

```
+CIPSERVER:<mode>[,<port>,<"type">,<CA enable>,<netif>]
OK
```

设置命令

命令:

```
AT+CIPSERVER=<mode>[,<param2>][,<"type">][,<CA enable>][,<netif>]
```

响应:

```
OK
```

参数

- **<mode>**:
 - 0: 关闭服务器
 - 1: 建立服务器
- **<param2>**: 参数 <mode> 不同, 则此参数意义不同:
 - 如果 <mode> 是 1, <param2> 代表端口号。默认值: 333
 - 如果 <mode> 是 0, <param2> 代表服务器是否关闭所有客户端。默认值: 0
 - * 0: 关闭服务器并保留现有客户端连接
 - * 1: 关闭服务器并关闭所有连接
- **<"type">**: 服务器类型: "TCP", "TCPv6", "SSL", 或 "SSLv6"。默认值: "TCP"。
- **<CA enable>**:
 - 0: 不使用 CA 认证
 - 1: 使用 CA 认证
- **<netif>**: 指定服务器监听的网络接口, 默认值: 0。
 - 0: 所有网络接口
 - 1: Wi-Fi station 接口
 - 2: Wi-Fi SoftAP 接口

说明

- 多连接情况下 ([AT+CIPMUX=1](#)), 才能开启服务器。
- 创建服务器后, 自动建立服务器监听, 最多只允许创建一个服务器。
- 当有客户端接入, 会自动占用一个连接 ID。
- 如果你想基于 IPv6 网络创建一个 TCP/SSL 服务器, 请首先设置 [AT+CIPV6=1](#), 并获取一个 IPv6 地址。
- 关闭服务器时, 参数 <"type">、<CA enable> 和 <netiif> 必须省略。
- 若需查看建立服务器失败原因, 请先运行 [AT+SYSLOG=1](#) 启用日志后重试本命令。失败时, AT 会返回更详细的错误码 +ERRNO:<error_code>, 以便定位问题。
- 关于 TLS 协议版本配置: 请参考 [修改 TLS 协议版本说明](#)。

示例

```
// 建立 TCP 服务器
AT+CIPMUX=1
AT+CIPSERVER=1,80

// 建立 SSL 服务器
AT+CIPMUX=1
AT+CIPSERVER=1,443,"SSL",1

// 基于 IPv6 网络，创建 SSL 服务器
AT+CIPMUX=1
AT+CIPSERVER=1,443,"SSLv6",0

// 关闭服务器并且关闭所有连接
AT+CIPSERVER=0,1
```

3.3.17 AT+CIPSERVERMAXCONN: 查询/设置服务器允许建立的最大连接数

查询命令

功能:

查询 TCP 或 SSL 服务器允许建立的最大连接数

命令:

```
AT+CIPSERVERMAXCONN?
```

响应:

```
+CIPSERVERMAXCONN:<num>
OK
```

设置命令

功能:

设置 TCP 或 SSL 服务器允许建立的最大连接数

命令:

```
AT+CIPSERVERMAXCONN=<num>
```

响应:

```
OK
```

参数

- **<num>**: TCP 或 SSL 服务器允许建立的最大连接数，范围：[1,5]。如果你想修改该参数的上限阈值，请参考 [AT+CIPSTART](#) 命令中参数 `<link ID>` 的描述。

说明

- 如需设置最大连接数 (AT+CIPSERVERMAXCONN=<num>)，请在创建服务器之前设置。

示例

```
AT+CIPMUX=1
AT+CIPSERVERMAXCONN=2
AT+CIPSERVER=1,80
```

3.3.18 AT+CIPMODE: 查询/设置传输模式

查询命令

功能:

查询传输模式

命令:

```
AT+CIPMODE?
```

响应:

```
+CIPMODE:<mode>
OK
```

设置命令

功能:

设置传输模式

命令:

```
AT+CIPMODE=<mode>
```

响应:

```
OK
```

参数

- **<mode>:**
 - 0: 普通传输模式
 - 1: Network 透传接收模式, 仅支持 TCP 单连接、UDP 固定通信对端、SSL 单连接的情况

说明

- 配置更改不保存到 flash。
- 在 ESP32-C6 进入 Network 透传接收模式后, 任何蓝牙功能将无法使用。

示例

```
AT+CIPMODE=1
```

3.3.19 AT+CIPSTO: 查询/设置本地 TCP/SSL 服务器超时时间

查询命令

功能:

查询本地 TCP/SSL 服务器超时时间

命令:

```
AT+CIPSTO?
```

响应:

```
+CIPSTO:<time>  
OK
```

设置命令

功能:

设置本地 TCP/SSL 服务器超时时间

命令:

```
AT+CIPSTO=<time>
```

响应:

```
OK
```

参数

- **<time>**: 本地 TCP/SSL 服务器超时时间, 单位: 秒, 取值范围: [0,7200]

说明

- 当 TCP/SSL 客户端在 <time> 时间内未发生数据通讯时, ESP32-C6 服务器会断开此连接。
- 如果设置参数 <time> 为 0, 则连接永远不会超时, 不建议这样设置。
- 在设定的时间内, 当客户端发起与服务器的通信或者服务器发起与客户端的通信时, 计时器将重新计时。超时后, 客户端被关闭。

示例

```
AT+CIPMUX=1  
AT+CIPSERVER=1,1001  
AT+CIPSTO=10
```

3.3.20 AT+CIPSNTPCFG: 查询/设置时区和 SNTP 服务器

查询命令

命令:

```
AT+CIPSNTPCFG?
```

响应:

```
+CIPSNTPCFG:<enable>,<timezone>[,<"SNTP server1">][,<"SNTP server2">][,<"SNTP
↪server3">]
OK
```

设置命令**命令:**

```
AT+CIPSNTPCFG=<enable>[,<timezone>][,<"SNTP server1">][,<"SNTP server2">][,<"SNTP
↪server3">]
```

响应:

```
OK
```

参数

- **<enable>**: 设置 SNTP 服务器:
 - 1: 设置 SNTP 服务器;
 - 0: 不设置 SNTP 服务器。
- **<timezone>**: 支持以下两种格式:
 - 第一种格式的范围: [-12,14], 它以小时为单位, 通过与协调世界时 (UTC) 的偏移来标记大多数时区 ([UTC-12:00](#) 至 [UTC+14:00](#));
 - 第二种格式为 UTC 偏移量, UTC 偏移量指定了你需加多少时间到 UTC 时间上才能得到本地时间, 通常显示为 [+|-][hh]mm。如果当地时区在本初子午线以西, 则为负数, 如果在东边, 则为正数。小时 (hh) 必须在 -12 到 14 之间, 分钟 (mm) 必须在 0 到 59 之间。例如, 如果你想把时区设置为新西兰查塔姆群岛, 即 UTC+12:45, 你应该把 <timezone> 参数设置为 1245, 更多信息请参考 [UTC 偏移量](#)。
- **<"SNTP server1">**: 第一个 SNTP 服务器。
- **<"SNTP server2">**: 第二个 SNTP 服务器。
- **<"SNTP server3">**: 第三个 SNTP 服务器。

说明

- 设置命令若未填写以上三个 SNTP 服务器参数, 则默认使用 “cn.ntp.org.cn”、“ntp.sjtu.edu.cn” 和 “us.pool.ntp.org” 其中之一。
- 对于查询命令, 查询的 <timezone> 参数可能会和设置的 <timezone> 参数不一样。因为 <timezone> 参数支持第二种 UTC 偏移量格式, 例如: 设置 AT+CIPSNTPCFG=1,015, 那么查询时, ESP-AT 会忽略时区参数的前导 0, 即设置值是 15。不属于第一种格式, 所以按照第二种 UTC 偏移量格式解析, 也就是 UTC+00:15, 也就是查询出来的是 0 时区。
- 由于 SNTP 是基于 UDP 协议发送请求和接收回复, 当网络丢包时, 会导致 ESP32-C6 的时间无法及时同步。一旦 AT 命令口输出 [+TIME_UPDATED](#), 代表时间已同步, 此时你可以发送 [AT+CIPSNTPTIME?](#) 命令查询当前时间。

示例

```
// 使能 SNTP 服务器, 设置中国时区 (UTC+08:00)
AT+CIPSNTPCFG=1,8,"cn.ntp.org.cn","ntp.sjtu.edu.cn"
或
AT+CIPSNTPCFG=1,800,"cn.ntp.org.cn","ntp.sjtu.edu.cn"

// 使能 SNTP 服务器, 设置美国纽约的时区 (UTC-05:00)
AT+CIPSNTPCFG=1,-5,"0.pool.ntp.org","time.google.com"
```

(续下页)

(接上页)

```
或
AT+CIPSNTPCFG=1,-500,"0.pool.ntp.org","time.google.com"

// 使能 SNTP 服务器，设置新西兰时区查塔姆群岛的时区 (Chatham Islands, UTC+12:45)
AT+CIPSNTPCFG=1,1245,"0.pool.ntp.org","time.google.com"
```

3.3.21 AT+CIPSNTPTIME: 查询 SNTP 时间

查询命令

命令:

```
AT+CIPSNTPTIME?
```

响应:

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

说明

- 有关 asctime 时间的定义请见 [asctime man page](#)。
- 在 ESP32-C6 进入 Light-sleep 或 Deep-sleep 后再唤醒，系统时间可能会不准。建议你重新发送 [AT+CIPSNTPCFG](#) 命令，从 NTP 服务器获取新的时间。
- SNTP 获取到的时间存储在 RTC 区域，因此在软重启（芯片不掉电）后，时间不会丢失。

示例

```
AT+CWMODE=1
AT+CWJAP="1234567890","1234567890"
AT+CIPSNTPCFG=1,8,"cn.ntp.org.cn","ntp.sjtu.edu.cn"
AT+CIPSNTPTIME?
+CIPSNTPTIME:Tue Oct 19 17:47:56 2021
OK

或

AT+CWMODE=1
AT+CWJAP="1234567890","1234567890"
AT+CIPSNTPCFG=1,530
AT+CIPSNTPTIME?
+CIPSNTPTIME:Tue Oct 19 15:17:56 2021
OK
```

3.3.22 AT+CIPSNTPTINTV: 查询/设置 SNTP 时间同步的间隔

查询命令

命令:

```
AT+CIPSNTPTINTV?
```

响应:


```
+CIPSNTPINTV:<interval second>
```

```
OK
```

设置命令

命令:

```
AT+CIPSNTPINTV=<interval second>
```

响应:

```
OK
```

参数

- **<interval second>**: SNTP 时间同步间隔。单位: 秒。范围: [15,4294967]。

说明

- 配置了时间同步间隔, 意味着 ESP32-C6 多久一次向 NTP 服务器获取新的时间。

示例

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

```
OK
```

```
// 每小时同步一次时间
```

```
AT+CIPSNTPINTV=3600
```

```
OK
```

3.3.23 AT+CIPFWVER: 查询服务器已有的 AT 固件版本

查询命令

功能:

查询服务器已有的 ESP32-C6 AT 固件版本

命令:

```
AT+CIPFWVER?
```

响应:

```
+CIPFWVER:<"version">
```

```
OK
```

参数

- **<" version" >**: ESP32-C6 AT 固件版本

说明

- 在选择要升级的 OTA 版本时，强烈不建议从高版本向低版本升级。

3.3.24 AT+CIUPDATE: 通过 Network 升级固件

ESP-AT 在运行时，通过 Network 从指定的服务器上下载新固件到某些分区，从而升级固件。

查询命令

功能:

查询 ESP32-C6 设备的升级状态

命令:

```
AT+CIUPDATE?
```

响应:

```
+CIPUPDATE:<state>
OK
```

执行命令

功能:

在阻塞模式下通过 OTA 升级到 TCP 服务器上最新版本的固件

命令:

```
AT+CIUPDATE
```

响应:

请参考设置命令中的[响应](#)

设置命令

功能:

升级到服务器上指定版本的固件

命令:

```
AT+CIUPDATE=<ota mode>[,<"version">][,<"firmware name">][,<nonblocking>]
```

响应:

如果 OTA 在阻塞模式下成功，返回：

```
+CIPUPDATE:1
+CIPUPDATE:2
+CIPUPDATE:3
+CIPUPDATE:4
OK
```

如果 OTA 在非阻塞模式下成功，返回：

```
OK
+CIPUPDATE:1
+CIPUPDATE:2
+CIPUPDATE:3
+CIPUPDATE:4
```

如果在阻塞模式下 OTA 失败，返回：

```
+CIPUPDATE:<state>

ERROR
```

如果在非阻塞模式下 OTA 失败，返回：

```
OK
+CIPUPDATE:<state>
+CIPUPDATE:-1
```

参数

- **<ota mode>**:
 - 0: 通过 HTTP OTA;
 - 1: 通过 HTTPS OTA, 如果无效, 请检查 `./build.py menuconfig>Component config`
`>AT>OTA based upon ssl` 是否使能, 更多信息请见 [本地编译 ESP-AT 工程](#)。
- **<version>**: AT 版本, 如 v1.2.0.0、v1.1.3.0 或 v1.1.2.0。
- **<"firmware name">**: 升级的固件, 如 ota、mqtt_ca、client_ca 或其它 at_customize.csv 中自定义的分区。
- **<nonblocking>**:
 - 0: 阻塞模式的 OTA。此模式下, 直到 OTA 升级成功或失败后才可以发送 AT 命令。
 - 1: 非阻塞模式的 OTA。此模式下, 升级完成后 (+CIPUPDATE:4) 需手动重启。
- **<state>**:
 - 1: 找到服务器;
 - 2: 连接至服务器;
 - 3: 获得升级版本;
 - 4: 完成升级;
 - -1: 非阻塞模式下 OTA 失败。

说明

- 升级速度取决于网络状况。
- 如果网络条件不佳导致升级失败, AT 将返回 ERROR, 请等待一段时间再试。
- 如果你直接使用乐鑫提供的 AT [BIN](#), 本命令将从 Espressif Cloud 下载 AT 固件升级。
- 如果你使用的是自行编译的 AT BIN, 请自行实现 AT+CIPUPDATE FOTA 功能或者使用 [AT+USEROTA](#) 或者 [AT+WEBSERVER](#) 命令, 可参考 ESP-AT 工程提供的示例 [FOTA](#)。
- 建议升级 AT 固件后, 调用 [AT+RESTORE](#) 恢复出厂设置。
- OTA 过程的超时时间为 3 分钟。
- 非阻塞模式响应中的 OK 和 +CIPUPDATE:<state> 在输出顺序上没有严格意义上的先后顺序。OK 可能在 +CIPUPDATE:<state> 之前输出, 也有可能在 +CIPUPDATE:<state> 之后输出。
- 不建议升级到旧版本。降到旧版本会存在一定的兼容性问题, 甚至无法运行, 如果你坚持要升级到旧版本, 请根据自己的产品自行测试验证功能。
- 请参考 [如何实现 OTA 升级](#) 获取更多 OTA 命令。

示例

```

AT+CWMODE=1
AT+CWJAP="1234567890","1234567890"
AT+CIUPDATE
AT+CIUPDATE=1
AT+CIUPDATE=1,"v1.2.0.0"
AT+CIUPDATE=1,"v2.2.0.0","mqtt_ca"
AT+CIUPDATE=1,"v2.2.0.0","ota",1
AT+CIUPDATE=1,,,1
AT+CIUPDATE=1,"ota",1
AT+CIUPDATE=1,"v2.2.0.0",,1

```

3.3.25 AT+CIPDINFO: 设置 +IPD 消息详情

查询命令

命令:

```
AT+CIPDINFO?
```

响应:

```
+CIPDINFO:true
OK
```

或

```
+CIPDINFO:false
OK
```

设置命令

命令:

```
AT+CIPDINFO=<mode>
```

响应:

```
OK
```

参数

- **<mode>**:
 - 0: 在“+IPD”和“+CIPRECVDATA”消息中，不提示对端 IP 地址和端口信息
 - 1: 在“+IPD”和“+CIPRECVDATA”消息中，提示对端 IP 地址和端口信息

示例

```
AT+CIPDINFO=1
```

3.3.26 AT+CIPSSLCONF: 查询/设置 SSL 客户端配置

查询命令

功能:

查询 ESP32-C6 作为 SSL 客户端时每个连接的配置信息

命令:

```
AT+CIPSSLCONF?
```

响应:

```
+CIPSSLCONF:<link ID>,<auth_mode>,<pki_number>,<ca_number>
OK
```

设置命令

命令:

```
// 单连接: (AT+CIPMUX=0)
AT+CIPSSLCONF=<auth_mode>[,<pki_number>][,<ca_number>]

// 多连接: (AT+CIPMUX=1)
AT+CIPSSLCONF=<link ID>,<auth_mode>[,<pki_number>][,<ca_number>]
```

响应:

```
OK
```

参数

- **<link ID>**: 网络连接 ID (0 ~ max), 在多连接的情况下, 若参数值设为 max, 则表示所有连接, 本参数默认值为 5。
- **<auth_mode>**:
 - 0: 不认证, 此时无需填写 <pki_number> 和 <ca_number> 参数;
 - 1: ESP-AT 提供客户端证书供服务器端 CA 证书校验;
 - 2: ESP-AT 客户端载入 CA 证书来校验服务器端的证书;
 - 3: 相互认证。
- **<pki_number>**: 证书和私钥的索引, 如果只有一个证书和私钥, 其值应为 0。
- **<ca_number>**: CA 的索引, 如果只有一个 CA, 其值应为 0。

说明

- 如果想要本配置立即生效, 请在建立 SSL 连接前运行本命令。
- 配置更改将保存在 NVS 区, 如果你使用 [AT+SAVETRANSLINK](#) 命令设置开机进入 SSL 透传模式, ESP32-C6 将在下次上电时基于本配置建立 SSL 连接。
- 如果你想使用自己的证书, 运行时请使用 [AT+SYSMF](#) 命令更新 SSL 证书 (具体步骤请参考 [AT+SYSMF 命令示例](#))。如果你想预烧录自己的证书, 请参考 [如何更新 PKI 配置](#)。
- 如果 <auth_mode> 配置为 2 或者 3, 为了校验服务器的证书有效期, 请在发送 [AT+CIPSTART](#) 命令前确保 ESP32-C6 已获取到当前时间。(你可以发送 [AT+CIPSNTPCFG](#) 命令来配置 SNTP, 获取当前时间, 发送 [AT+CIPSNTPTIME?](#) 命令查询当前时间)。

3.3.27 AT+CIPSSLCCIPHER: 查询/设置 SSL 客户端的密码套件 (Cipher Suite)

查询命令

功能:

查询 ESP32-C6 作为 SSL 客户端时支持的密码套件

命令:

```
AT+CIPSSLCCIPHER?
```

响应:

```
+CIPSSLCCIPHER:<idx>,<cipher_suite>
OK
```

设置命令

命令:

```
// 单连接: (AT+CIPMUX=0)
AT+CIPSSLCCIPHER=<counts>[,<cipher_suite>][...][,<cipher_suite>]

// 多连接: (AT+CIPMUX=1)
AT+CIPSSLCCIPHER=<link ID>,<counts>[,<cipher_suite>][...][,<cipher_suite>]
```

响应:

```
OK
```

参数

- **<idx>**: 密码套件的索引号, 从 0 开始。
- **<link ID>**: 网络连接 ID (0 ~ max), 在多连接的情况下, 若参数值设为 max, 则表示所有连接, 本参数默认值为 5。
- **<counts>**: 密码套件的数量。最大值受限于命令的最大长度 256 字节。
 - 0: 清除已配置的密码套件。
 - 其他值: 设置密码套件的个数。
- **<cipher_suite>**: 表示 ClientHello 中的密码套件。对应的值定义在 [ssl_ciphersuites.h](#) 中。

说明

- 如果想要本配置立即生效, 请在建立 SSL 连接前运行本命令。

示例

```
// 单连接: (AT+CIPMUX=0), 密码套件为 TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 和
↪ TLS_ECDHE_ECDSA_WITH_AES_256_GCM
AT+CIPSSLCCIPHER=2,0xC023,0xC0AD

// 多连接: (AT+CIPMUX=1), 密码套件为 TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 和
↪ TLS_ECDHE_ECDSA_WITH_AES_256_GCM
AT+CIPSSLCCIPHER=0,2,0xC023,0xC0AD
```

3.3.28 AT+CIPSSLCCN: 查询/设置 SSL 客户端的公用名 (common name)

查询命令

功能:

查询每个 SSL 连接中客户端的通用名称

命令:

```
AT+CIPSSLCCN?
```

响应:

```
+CIPSSLCCN:<link ID>,<"common name">
OK
```

设置命令

命令:

```
// 单连接: (AT+CIPMUX=0)
AT+CIPSSLCCN=<"common name">

// 多连接: (AT+CIPMUX=1)
AT+CIPSSLCCN=<link ID>,<"common name">
```

响应:

```
OK
```

参数

- **<link ID>**: 网络连接 ID (0 ~ max), 在单连接的情况下, 本参数值为 0; 在多连接的情况下, 若参数值设为 max, 则表示所有连接; 本参数默认值为 5。
- **<"common name">**: 本参数用来认证服务器发送的证书中的公用名。

说明

- 如果想要本配置立即生效, 请在建立 SSL 连接前运行本命令。

3.3.29 AT+CIPSSLCSNI: 查询/设置 SSL 客户端的 SNI

查询命令

功能:

查询每个连接的 SNI 配置

命令:

```
AT+CIPSSLCSNI?
```

响应:

```
+CIPSSLCSNI:<link ID>,<"sni">
OK
```

设置命令

命令:

单连接：(AT+CIPMUX=0)
AT+CIPSSLCSNI=<"sni">

多连接：(AT+CIPMUX=1)
AT+CIPSSLCSNI=<link ID>,<"sni">

响应：

OK

参数

- <link ID>：网络连接 ID (0 ~ max)，在单连接的情况下，本参数值为 0；在多连接的情况下，若参数值设为 max，则表示所有连接；本参数默认值为 5。
- <"sni">：ClientHello 里的 SNI。

说明

- 如果想要本配置立即生效，请在建立 SSL 连接前运行本命令。

3.3.30 AT+CIPSSLCALPN：查询/设置 SSL 客户端 ALPN

查询命令

功能：

查询 ESP32-C6 作为 SSL 客户端时每个连接的 ALPN 配置

命令：

AT+CIPSSLCALPN?

响应：

+CIPSSLCALPN:<link ID>,<"alpn">[,<"alpn">][,<"alpn">]

OK

设置命令

命令：

```
// 单连接：(AT+CIPMUX=0)
AT+CIPSSLCALPN=<counts>[,<"alpn">][,<"alpn">][,<"alpn">]

// 多连接：(AT+CIPMUX=1)
AT+CIPSSLCALPN=<link ID>,<counts>[,<"alpn">][,<"alpn">[,<"alpn">]
```

响应：

OK

参数

- **<link ID>**: 网络连接 ID (0 ~ max), 在单连接的情况下, 本参数值为 0; 在多连接的情况下, 若参数值设为 max, 则表示所有连接; 本参数默认值为 5。
- **<counts>**: ALPN 的数量。范围: [0,5]。
 - 0: 清除 ALPN 配置。
 - [1,5]: 设置 ALPN 配置。
- **<" alpn" >**: 字符串参数, 表示 ClientHello 中的 ALPN。ALPN 最大长度受限于命令的最大长度。

说明

- 如果想要本配置立即生效, 请在建立 SSL 连接前运行本命令。

3.3.31 AT+CIPSSLCPSK: 查询/设置 SSL 客户端的 PSK (字符串格式)

查询命令

功能:

查询 ESP32-C6 作为 SSL 客户端时每个连接的 PSK 配置

命令:

```
AT+CIPSSLCPSK?
```

响应:

```
+CIPSSLCPSK:<link ID>,<"psk">,<"hint">
OK
```

设置命令

命令:

```
// 单连接: (AT+CIPMUX=0)
AT+CIPSSLCPSK=<"psk">,<"hint">

// 多连接: (AT+CIPMUX=1)
AT+CIPSSLCPSK=<link ID>,<"psk">,<"hint">
```

响应:

```
OK
```

参数

- **<link ID>**: 网络连接 ID (0 ~ max), 在单连接的情况下, 本参数值为 0; 在多连接的情况下, 若参数值设为 max, 则表示所有连接; 本参数默认值为 5。
- **<" psk" >**: PSK identity (字符串格式), 最大长度: 32。如果你的 <"psk"> 参数包含 \0, 请使用 [AT+CIPSSLCPSKHEX](#) 命令。
- **<" hint" >**: PSK hint, 最大长度: 32。

说明

- 如果想要本配置立即生效, 请在建立 SSL 连接前运行本命令。

3.3.32 AT+CIPSSLCPKHEX: 查询/设置 SSL 客户端的 PSK (十六进制格式)

说明

- 类似于 [AT+CIPSSLCPK](#) 命令，该命令也用于设置或查询 SSL 客户端的预共享密钥 (PSK)，但其 <"psk"> 参数使用十六进制格式而不是字符串格式。因此，<"psk"> 参数中的 \0 表示为 00。

示例

```
// 单连接: (AT+CIPMUX=0), PSK identity 为 "psk", PSK hint 为 "myhint"
AT+CIPSSLCPKHEX="70736b", "myhint"

// 多连接: (AT+CIPMUX=1), PSK identity 为 "psk", PSK hint 为 "myhint"
AT+CIPSSLCPKHEX=0, "70736b", "myhint"
```

3.3.33 AT+CIPRECONNINTV: 查询/设置 Network 透传模式下的 TCP/UDP/SSL 重连间隔

查询命令

功能:

查询 Network [透传模式](#) 下的自动重连间隔

命令:

```
AT+CIPRECONNINTV?
```

响应:

```
+CIPRECONNINTV:<interval>
OK
```

设置命令

功能:

设置 Network [透传模式](#) 下 TCP/UDP/SSL 传输断开后自动重连的间隔

命令:

```
AT+CIPRECONNINTV=<interval>
```

响应:

```
OK
```

参数

- <interval>: 自动重连间隔时间，单位：100 毫秒，默认值：1，范围：[1,36000]。

说明

- 若 [AT+SYSTORE=1](#) 时，配置更改将保存在 NVS 区。

示例

```
AT+CIPRECONNINTV=10
```

3.3.34 AT+CIPRECVMODE: 查询/设置套接字接收模式

查询命令

功能:

查询套接字接收模式

命令:

```
AT+CIPRECVMODE?
```

响应:

```
+CIPRECVMODE:<link ID>,<mode>
OK
```

设置命令

命令:

```
// 单连接: (AT+CIPMUX=0)
AT+CIPRECVMODE=<mode>[,<data_len_report_mode>]

// 多连接: (AT+CIPMUX=1)
AT+CIPRECVMODE=<link ID>,<mode>[,<data_len_report_mode>]
```

响应:

```
OK
```

参数

- **<link ID>**: 网络连接 ID (0 ~ max), 在单连接的情况下, 本参数值为 0; 在多连接的情况下, 若参数值设为 max, 则表示所有连接; 本参数默认值为 5。
- **<mode>**: 套接字数据接收模式, 默认值: 0。
 - 0: 主动模式, ESP-AT 将所有接收到的套接字数据立即发送给主机 MCU, 头为 “+IPD” (套接字接收窗口为 5760 字节, 每次向 MCU 最大发送 2920 字节有效数据)。
 - 1: 被动模式, ESP-AT 将所有接收到的套接字数据保存到内部缓存区 (套接字接收窗口, 默认值为 5760 字节), 等待 MCU 读取。对于 TCP 和 SSL 连接, 如果缓存区满了, 将阻止套接字传输; 对于 UDP 传输, 如果缓存区满了, 则会发生数据丢失。
- **<data_len_report_mode>**: 数据长度提示模式, 默认值: 0。
 - 0: +IPD 和 +CIPRECLEN 消息中上报的长度为当前接收缓冲区中套接字数据的总长度。
 - 1: +IPD 和 +CIPRECLEN 消息中上报的长度为下一个数据报文的长度。在 UDP 传输中, 配置为 1 时, 可以确保每次上报的长度都对应一个完整的数据报文。

说明

- 该配置不能用于 Network 透传模式。
- 当 ESP-AT 在被动模式下收到套接字数据时, 会根据情况的不同提示不同的信息:
 - 多连接时 (AT+CIPMUX=1), 提示 +IPD,<link ID>,<len>;

- 单连接时 (AT+CIPMUX=0), 提示 +IPD, <len>。
- <len> 表示缓存区中套接字数据的总长度。
- 一旦有 +IPD 报出, 应该运行 **AT+CIPRECVDATA** 来读取数据。否则, 在前一个 +IPD 被读取之前, 下一个 +IPD 将不会被报告给主机 MCU。
- 在断开连接的情况下, 缓冲的套接字数据仍然存在, MCU 仍然可以读取, 直到发送 **AT+CIPCLOSE** (AT 作为客户端) 或 **AT+CIPSERVER=0,1** (AT 作为服务器)。换句话说, 如果 +IPD 已经被报告, 那么在你发送 **AT+CIPCLOSE** 或发送 **AT+CIPSERVER=0,1** 或通过 **AT+CIPRECVDATA** 命令读取所有数据之前, 这个连接的 CLOSED 信息永远不会出现。
- 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 [示例](#), 使用被动接收数据模式。

示例

```
// 单连接模式下, 设置被动接收模式
AT+CIPRECVMODE=1

// 多连接模式下, 设置所有连接为被动接收模式
AT+CIPRECVMODE=5,1
```

3.3.35 AT+CIPRECVDATA: 获取被动接收模式下的套接字数据

设置命令

命令:

```
// 单连接: (AT+CIPMUX=0)
AT+CIPRECVDATA=<len>[,<keep_datagram_integrity>]

// 多连接: (AT+CIPMUX=1)
AT+CIPRECVDATA=<link_id>,<len>[,<keep_datagram_integrity>]
```

响应:

```
+CIPRECVDATA:<actual_len>,<data>
OK
```

或

```
+CIPRECVDATA:<actual_len>,<"remote IP">,<remote port>,<data>
OK
```

参数

- **<link_id>**: 多连接模式下的连接 ID。
- **<len>**: 最大值为: 0x7fffff, 如果实际收到的数据长度比本参数值小, 则返回实际长度的数据。
- **<keep_datagram_integrity>**: 保持数据报完整性标志, 默认值: 0。
 - 0: 不保持数据报完整性, ESP-AT 将尽可能多地返回数据, 直到达到 <len> 参数指定的长度。
 - 1: 保持数据报完整性, ESP-AT 将确保每次返回的数据都是一个完整的数据报文, 且不超过 <len> 参数指定的长度。
- **<actual_len>**: 实际获取的数据长度。
- **<data>**: 获取的数据。
- **<"remote IP">**: 字符串参数, 表示对端 IP 地址, 通过 **AT+CIPDINFO=1** 命令使能。
- **<remote port>**: 对端端口, 通过 **AT+CIPDINFO=1** 命令使能。

说明

- 该命令需要在被动接收模式下执行，否则会直接返回 **ERROR**，可以通过 [AT+CIPRCVTYPE?](#) 命令确认是否是在被动接收模式。
- 该命令在没有数据可读的情况下执行时会直接返回 **ERROR**，可以通过 [AT+CIPRCVLEN?](#) 命令确认此时是否有可读数据。
- 执行 `AT+CIPRCVDATA=<len>` 命令时，至少需要 `<len> + 128` 字节的内存，你可以使用命令 [AT+SYSRAM?](#) 查询当前可用内存情况。当内存不足导致内存申请失败时此命令也会返回 **ERROR**。你可以通过 [AT 输出日志口](#) 查看是否有类似 `alloc fail` 的打印信息，以确认是否出现了内存分配失败的情况。

示例

```
AT+CIPRCVTYPE=1

// 例如，如果主机 MCU 从 0 号连接中收到 100 字节的数据，
// 则会提示消息 "+IPD,0,100"，
// 然后，你可以通过运行以下命令读取这 100 字节的数据：
AT+CIPRCVDATA=0,100
```

3.3.36 AT+CIPRCVLEN：查询被动接收模式下套接字数据的长度

查询命令

功能：

查询某一连接中缓存的所有数据长度

命令：

```
AT+CIPRCVLEN?
```

响应：

```
+CIPRCVLEN:<data length of link>[...][,<data length of link>]

OK
```

参数

- <data length of link>**：某一连接中缓冲的所有数据长度。

说明

- SSL 连接中，ESP-AT 返回的数据长度可能会小于真实数据的长度。

示例

```
AT+CIPRCVLEN?
+CIPRCVLEN:100,,,,,
OK
```

3.3.37 AT+PING: ping 对端主机

设置命令

功能:

ping 对端主机

命令:

```
AT+PING=<"host">
```

响应:

```
+PING:<time>
```

```
OK
```

或

```
+PING:TIMEOUT // 只有在域名解析失败或 PING 超时情况下，才会有这个回复
```

```
ERROR
```

参数

- <"host">: 字符串参数，表示对端主机的 IPv4 地址，IPv6 地址，或域名。
- <time>: ping 的响应时间，单位：毫秒。

说明

- 如果你想基于 IPv6 网络 Ping 对端主机，请执行以下操作：
 - 确保 AP 支持 IPv6
 - 设置 [AT+CIPV6=1](#)
 - 通过 [AT+CWJAP](#) 命令获取到一个 IPv6 地址
 - （可选）通过 [AT+CIPSTA?](#) 命令检查 ESP32-C6 是否获取到 IPv6 地址
- 如果远端主机是域名字符串，则 ping 将先通过 DNS 进行域名解析（优先解析 IPv4 地址），再 ping 对端主机 IP 地址

示例

```
AT+PING="192.168.1.1"
AT+PING="www.baidu.com"

// 下一代互联网国家工程中心
AT+PING="240c::6666"
```

3.3.38 AT+CIPDNS: 查询/设置 DNS 服务器信息

查询命令

功能:

查询当前 DNS 服务器信息

命令:

```
AT+CIPDNS?
```

响应:

```
+CIPDNS:<enable>[,<"DNS IP1">][,<"DNS IP2">][,<"DNS IP3">]
OK
```

设置命令

功能:

设置 DNS 服务器信息

命令:

```
AT+CIPDNS=<enable>[,<"DNS IP1">][,<"DNS IP2">][,<"DNS IP3">]
```

响应:

```
OK
```

或

```
ERROR
```

参数

- **<enable>**: 设置 DNS 服务器
 - 0: 启用自动获取 DNS 服务器设置, DNS 服务器将会恢复为 208.67.222.222、114.114.114.114 和 8.8.8.8, 只有当 ESP32-C6 station 完成了 DHCP 过程, DNS 服务器才有可能更新。
 - 1: 启用手动设置 DNS 服务器信息, 如果不设置参数 <DNS IPx> 的值, 则使用默认值 208.67.222.222、114.114.114.114 和 8.8.8.8。
- **<"DNS IP1">**: 第一个 DNS 服务器 IP 地址, 对于设置命令, 只有当 <enable> 参数为 1 时, 也就是启用手动 DNS 设置, 本参数才有效; 如果设置 <enable> 为 1, 并为本参数设置一个值, 当你运行查询命令时, ESP-AT 将把该参数作为当前的 DNS 设置返回。
- **<"DNS IP2">**: 第二个 DNS 服务器 IP 地址, 对于设置命令, 只有当 <enable> 参数为 1 时, 也就是启用手动 DNS 设置, 本参数才有效; 如果设置 <enable> 为 1, 并为本参数设置一个值, 当你运行查询命令时, ESP-AT 将把该参数作为当前的 DNS 设置返回。
- **<"DNS IP3">**: 第三个 DNS 服务器 IP 地址, 对于设置命令, 只有当 <enable> 参数为 1 时, 也就是启用手动 DNS 设置, 本参数才有效; 如果设置 <enable> 为 1, 并为本参数设置一个值, 当你运行查询命令时, ESP-AT 将把该参数作为当前的 DNS 设置返回。

说明

- 若 **AT+SYSTORE=1**, 配置更改将保存在 NVS 区。
- 这三个参数不能设置在同一个服务器上。
- 当 <enable> 为 0 时, DNS 服务器可能会根据 ESP32-C6 设备所连接的路由器的配置而改变。
- 当 <enable> 为 1 时, 手动设置的 DNS 服务器将始终生效, DHCP 从路由器获取的 DNS 配置不会覆盖手动设置的值。若要恢复使用 DHCP 分配的 DNS, 需执行 AT+CIPDNS=0 重新启用自动获取。

示例

```
AT+CIPDNS=0
AT+CIPDNS=1,"208.67.222.222","114.114.114.114","8.8.8.8"

// 第一个基于 IPv6 的 DNS 服务器：下一代互联网国家工程中心
// 第二个基于 IPv6 的 DNS 服务器：google-public-dns-a.google.com
// 第三个基于 IPv6 的 DNS 服务器：江苏省主 DNS 服务器
AT+CIPDNS=1,"240c::6666","2001:4860:4860::8888","240e:5a::6666"
```

3.3.39 AT+MDNS：设置 mDNS 功能

- 设备开启一个 *mDNS* 服务
- 查询局域网里的 *mDNS* 服务

设备开启一个 mDNS 服务

设置命令 命令：

```
AT+MDNS=<mode>[,<"hostname">,<"service_type">,<port>][,<"instance">][,<"proto">][,
↪<txt_number>][,<"key">,<"value">][...]
```

响应：

```
OK
```

参数

- **<mode>**:
 - 1: 开启 mDNS 服务，后续参数需要填写
 - 0: 关闭 mDNS 功能，后续参数无需填写
- **<"hostname">**: mDNS 主机名称。
- **<"service_type">**: mDNS 服务类型。
- **<port>**: mDNS 服务端口。
- **<"instance">**: mDNS 实例名称。默认值: <"hostname">。
- **<"proto">**: mDNS 服务协议。建议值: `_tcp` 或 `_udp`，默认值: `_tcp`。
- **<txt_number>**: mDNS TXT 记录的数量。范围: [1,20]。
- **<"key">**: TXT 记录的键。
- **<"value">**: TXT 记录的值。
- **[...]**: 根据 <txt_number> 继续填写 TXT 记录的键值对。

示例

```
// 开启 mDNS 服务，主机名为 "espressif"，服务类型为 "_iot"，端口为 8080
AT+MDNS=1,"espressif","_iot",8080

// 关闭 mDNS 功能
AT+MDNS=0
```

详细示例参考[mDNS 服务示例](#)。

查询局域网里的 mDNS 服务

设置命令 命令：


```
// 查询局域网里的 PTR 记录
AT+MDNS=<mode>,<type>,<"service">,<"proto">[,<timeout>][,<max_results>]

// 查询局域网里的 TXT、SRV、TXT + SRV 记录
AT+MDNS=<mode>,<type>,<"instance">,<"service">,<"proto">[,<timeout>][,<max_results>
↪]

// 查询局域网里的 A、AAAA、A + AAAA 记录
AT+MDNS=<mode>,<type>,<"hostname">[,<timeout>][,<max_results>]
```

响应:

```
+MDNS:<data_len>,
IF=<netif>
IP=<ip_type>
PTR=<instance>
SRV=<hostname>:<port>
TXT=<key>=<value>
A=<ip4>
AAAA=<ip6>

OK
```

命令参数

- **<mode>**:
 - 2: 查询局域网里的 mDNS 服务，后续参数需要填写
 - 0: 关闭 mDNS 功能，后续参数无需填写
- **<type>**: 查询类型。
 - 0: PTR 记录
 - 1: TXT 记录
 - 2: SRV 记录
 - 3: TXT + SRV 记录
 - 4: A 记录
 - 5: AAAA 记录
 - 6: A + AAAA 记录
- **<" instance" >**: mDNS 实例名称。最大长度: 64 字节，超过会自动截断。
- **<" hostname" >**: 主机名称。最大长度: 64 字节，超过会自动截断。
- **<" service" >**: mDNS 服务类型。最大长度: 64 字节，超过会自动截断。
- **<" proto" >**: mDNS 服务协议。最大长度: 64 字节，超过会自动截断。建议值: `_tcp` 或 `_udp`。
- **<timeout>**: 查询超时时间，单位: 毫秒，默认值: 5000。范围: [1000,180000]。如果查询的个数未达到 `<max_results>`，则在超时后返回查询结果。
- **<max_results>**: 查询结果的最大数量，默认值: 1。范围: [1,255]。如果查询的个数未达到 `<max_results>`，则在超时后返回查询结果。否则，立即返回查询结果。

命令响应参数

- **<data_len>**: 查询结果的长度。
- **<netif>**: 网络接口，表示 mDNS 服务所在的网络接口。
 - 1: Wi-Fi Station 接口
 - 2: Wi-Fi SoftAP 接口
- **<ip_type>**: 网络接口的 IP 地址类型。
 - 0: IPv4 地址
 - 1: IPv6 地址
- **<instance>**: mDNS 实例名称。
- **<hostname>**: 主机名称。
- **<port>**: mDNS 服务端口。
- **<key>**: TXT 记录的键。

- **<value>**: TXT 记录的值。
- **<ip4>**: mDNS 服务所在的 IPv4 地址。
- **<ip6>**: mDNS 服务所在的 IPv6 地址。

示例

```
// PC 上通过 avahi-publish 发布 mDNS 服务
avahi-publish -s my_instance _my_printer._tcp 35 "version=v4.1.0.0"

// 通过 AT 命令查询局域网里的 PTR 记录
AT+MDNS=2,0,"_my_printer","_tcp"

// 响应结果
+MDNS:234,
IF=1
IP=0
PTR=my_instance
SRV=espressif-1.local:35
TXT=version=v4.1.0.0
A=192.168.200.249
AAAA=240e:1234:204c:bd02:d80f:8c45:3576:2574
AAAA=240e:1234:204c:bd02:f043:47f0:7c39:0002
AAAA=240e:1234:204c:bd02:2971:f412:14c6:fa00

OK

// 关闭 mDNS 功能
AT+MDNS=0
```

3.3.40 AT+CIPTCPOPT: 查询/设置套接字选项

查询命令

功能:

查询当前套接字选项

命令:

```
AT+CIPTCPOPT?
```

响应:

```
+CIPTCPOPT:<link_id>,<so_linger>,<tcp_nodelay>,<so_sndtimeo>,<keep_alive>
OK
```

设置命令

命令:

```
// 单连接: (AT+CIPMUX=0):
AT+CIPTCPOPT=[<so_linger>],[<tcp_nodelay>],[<so_sndtimeo>][,<keep_alive>]

// 多连接: (AT+CIPMUX=1):
AT+CIPTCPOPT=<link ID>,[<so_linger>],[<tcp_nodelay>],[<so_sndtimeo>][,<keep_alive>]
```

响应:

```
OK
```

或

ERROR

参数

- **<link_id>**: 网络连接 ID (0 ~ max), 在多连接的情况下, 若参数值设为 max, 则表示所有连接; 本参数默认值为 5。
- **<so_linger>**: 配置套接字的 SO_LINGER 选项 (参考: [SO_LINGER 介绍](#)), 单位: 秒, 默认值: -1。
 - = -1: 关闭;
 - = 0: 开启, linger time = 0;
 - > 0: 开启, linger time = <so_linger>;
- **<tcp_nodelay>**: 配置套接字的 TCP_NODELAY 选项 (参考: [TCP_NODELAY 介绍](#)), 默认值: 0。
 - 0: 禁用 TCP_NODELAY
 - 1: 启用 TCP_NODELAY
- **<so_sndtimeo>**: 配置套接字的 SO_SNDTIMEO 选项 (参考: [SO_SNDTIMEO 介绍](#)), 单位: 毫秒, 默认值: 0。
- **<keep_alive>**: 配置套接字的 SO_KEEPALIVE 选项 (参考: [SO_KEEPALIVE 介绍](#)), 单位: 秒。
 - 范围: [0,7200]。
 - * 0: 禁用 keep-alive 功能; (默认)。
 - * 1 ~ 7200: 开启 keep-alive 功能。TCP_KEEPIDLE 值为 <keep_alive>, TCP_KEEPINTVL 值为 1, TCP_KEEPCNT 值为 3。
 - 本命令中的 <keep_alive> 参数与 [AT+CIPSTART](#) 命令中的 <keep_alive> 参数相同, 最终值由后设置的命令决定。如果运行本命令时不设置 <keep_alive> 参数, 则默认使用上次配置的值。

说明

- 若 [AT+SYSTORE=1](#), 配置更改将保存到 NVS 分区。
- 在配置套接字选项前, 请充分了解该选项功能, 以及配置后可能的影响。
- SO_LINGER 选项不建议配置较大的值。例如配置 SO_LINGER 值为 60, 则 [AT+CIPCLOSE](#) 命令在收不到对端 TCP FIN 包情况下, 会导致 AT 阻塞 60 秒, 从而无法响应其它命令。因此, SO_LINGER 建议保持默认值。
- TCP_NODELAY 选项适用于吞吐量小但对实时性要求高的场景。开启后, [LwIP](#) 会加快 TCP 的发送, 但如果网络环境较差, 会由于重传而导致吞吐降低。因此, TCP_NODELAY 建议保持默认值。
- SO_SNDTIMEO 选项适用于 [AT+CIPSTART](#) 命令未配置 keepalive 参数的应用场景。配置本选项后, [AT+CIPSEND](#)、[AT+CIPSENDL](#)、[AT+CIPSENDEX](#) 命令将会在该超时内退出, 无论是否发送成功。这里, SO_SNDTIMEO 建议配置为 5 ~ 10 秒。
- SO_KEEPALIVE 选项适用于主动定时检测连接是否断开的场景, 通常 AT 作为 TCP 服务器时建议配置该选项。配置本选项后, 会增加额外的网络带宽。SO_KEEPALIVE 建议配置值不小于 60 秒。

3.4 Bluetooth® Low Energy AT 命令集

- [介绍](#)
- [AT+BLEINIT](#): Bluetooth LE 初始化
- [AT+BLEADDR](#): 查询/设置 Bluetooth LE 设备地址
- [AT+BLENAME](#): 查询/设置 Bluetooth LE 设备名称
- [AT+BLESCANPARAM](#): 查询/设置 Bluetooth LE 扫描参数
- [AT+BLESCAN](#): 使能 Bluetooth LE 扫描
- [AT+BLESCANRSPDATA](#): 设置 Bluetooth LE 扫描响应
- [AT+BLEADVPARAM](#): 查询/设置 Bluetooth LE 广播参数

- **AT+BLEADVDATA**: 设置 Bluetooth LE 广播数据
- **AT+BLEADVDATAEX**: 自动设置 Bluetooth LE 广播数据
- **AT+BLEADVSTART**: 开始 Bluetooth LE 广播
- **AT+BLEADVSTOP**: 停止 Bluetooth LE 广播
- **AT+BLECONN**: 建立 Bluetooth LE 连接
- **AT+BLECONNPARAM**: 查询/更新 Bluetooth LE 连接参数
- **AT+BLEDISCONN**: 断开 Bluetooth LE 连接
- **AT+BLECFGMTU**: 设置 Bluetooth LE MTU 长度
- **AT+BLEGATTSSRV**: GATTS 发现服务
- **AT+BLEGATTSCCHAR**: GATTS 发现服务特征
- **AT+BLEGATTSENTFY**: 服务器 notify 服务特征值给客户端
- **AT+BLEGATTSEND**: 服务器 indicate 服务特征值给客户端
- **AT+BLEGATTSETATTR**: GATTS 设置服务特征值
- **AT+BLEGATTCPRIMSRV**: GATTC 发现基本服务
- **AT+BLEGATTCINCLSRV**: GATTC 发现包含的服务
- **AT+BLEGATTCCHAR**: GATTC 发现服务特征
- **AT+BLEGATTCRD**: GATTC 读取服务特征值
- **AT+BLEGATTCWR**: GATTC 写服务特征值
- **AT+BLESPPCFG**: 查询/设置 Bluetooth LE SPP 参数
- **AT+BLESPP**: 进入 Bluetooth LE SPP 模式
- **AT+SAVETRANSLINK**: 设置 Bluetooth LE 开机透传模式信息
- **AT+BLESECPARAM**: 查询/设置 Bluetooth LE 加密参数
- **AT+BLEENC**: 发起 Bluetooth LE 加密请求
- **AT+BLEKEYREPLY**: 给对方设备回复密钥
- **AT+BLECONFREPLY**: 给对方设备回复确认结果（传统连接阶段）
- **AT+BLEENCDEV**: 查询绑定的 Bluetooth LE 加密设备列表
- **AT+BLEENCCLEAR**: 清除 Bluetooth LE 加密设备列表
- **AT+BLESETKEY**: 设置 Bluetooth LE 静态配对密钥
- **AT+BLUFI**: 开启或关闭 BluFi
- **AT+BLUFINAME**: 查询/设置 BluFi 设备名称
- **AT+BLUFISEND**: 发送 BluFi 用户自定义数据
- **AT+BLEPERIODICDATA**: 设置 Bluetooth LE 周期性广播数据
- **AT+BLEPERIODICSTART**: 开启 Bluetooth LE 周期性广播
- **AT+BLEPERIODICSTOP**: 停止 Bluetooth LE 周期性广播
- **AT+BLESYNCSTART**: 开启周期性广播同步
- **AT+BLESYNCSTOP**: 停止周期性广播同步
- **AT+BLEREADPHY**: 查询当前连接使用的 PHY
- **AT+BLESETPHY**: 设置当前连接使用的 PHY
- **AT+BLEOTANAME**: 查询/设置 Bluetooth LE OTA 设备名称
- **AT+BLEOTA**: 初始化或反初始化 Bluetooth LE OTA

3.4.1 介绍

当前，ESP32-C6 AT 固件支持 [蓝牙核心规范 5.0 版本](#)。

重要：默认的 AT 固件支持此页面下的所有 AT 命令。如果你需要修改 ESP32-C6 默认支持的命令，请自行编译 [ESP-AT 工程](#)，在第五步配置工程里选择（下面每项是独立的，根据你的需要选择）：

- 禁用 BluFi 命令：Component config>AT>AT blufi command support
- 禁用 Bluetooth LE 命令：Component config>AT>AT ble command support
- 禁用 Bluetooth LE HID 命令：Component config>AT>AT ble hid command support

3.4.2 AT+BLEINIT: Bluetooth LE 初始化

查询命令

功能:

查询 Bluetooth LE 是否初始化

命令:

```
AT+BLEINIT?
```

响应:

若已初始化, AT 返回:

```
+BLEINIT:<role>
OK
```

若未初始化, AT 返回:

```
+BLEINIT:0
OK
```

设置命令

功能:

设置 Bluetooth LE 初始化角色

命令:

```
AT+BLEINIT=<init>
```

响应:

```
OK
```

参数

- **<init>:**
- 0: 注销 Bluetooth LE
- 1: client 角色
- 2: server 角色
- 3: 双角色 (client 和 server)

说明

- 为获得更好的性能, 建议在使用 Bluetooth LE 功能前, 先发送 **AT+CWMODE=0/1** 命令禁用 SoftAP。如你了解更多细节, 请阅读 [RF 共存](#) 文档。
- 使用其它 Bluetooth LE 命令之前, 请先调用本命令, 初始化 Bluetooth LE 角色。
- Bluetooth LE 角色初始化后, 不能直接切换角色。如需切换角色, 请先执行 **AT+BLEINIT=0** 注销 Bluetooth LE, 或执行 **AT+RST** 重启系统, 然后重新初始化 Bluetooth LE 角色。
- 建议在注销 Bluetooth LE 之前, 停止正在进行的广播、扫描并断开所有的连接。
- 如果 Bluetooth LE 已初始化, 则 **AT+CIPMODE** 无法设置为 1。
- 如果设置为双角色, 则既可以使用服务器角色相关的命令, 也可以使用客户端角色相关的命令。
- 在双角色模式下, BLE 透传相关命令和 BLE 自动透传开机功能不支持。
- 在双角色模式下, 可以设置广播参数、查询广播参数并开始或停止广播。
- 在双角色模式下, 可以设置扫描参数、查询扫描参数并开始或停止扫描。

- 在双角色模式下，可以发起所有连接相关参数更新并查询连接参数。
- 在双角色模式下，连接和广播的总数限制为 3。
- 服务创建说明：当初始化 BLE 为 server 角色（<init> 为 2）或双角色（<init> 为 3）时，会自动创建 GATT 服务，无需单独的 AT 命令创建服务。

示例

```
AT+BLEINIT=1
```

3.4.3 AT+BLEADDR: 设置 Bluetooth LE 设备地址

查询命令

功能:

查询 Bluetooth LE 设备的随机地址

命令:

```
AT+BLEADDR?
```

响应:

```
+BLEADDR:<BLE_public_addr>  
OK
```

设置命令

功能:

设置 Bluetooth LE 设备的地址类型

命令:

```
AT+BLEADDR=<addr_type>[,<random_addr>]
```

响应:

```
OK
```

参数

- **<addr_type>**:
 - 0: 公共地址 (Public Address)
 - 1: 随机地址 (Random Address)

说明

- 静态地址 (Static Address) 应满足以下条件:
 - 地址最高两位应为 1;
 - 随机地址部分至少有 1 位为 0;
 - 随机地址部分至少有 1 位为 1。
- 默认情况下，查询命令返回公共地址。如果设置过随机地址，则查询命令返回随机地址。
- 设置的静态地址不会被保存在 NVS 区。

示例

```
AT+BLEADDR=1,"f8:7f:24:87:1c:7b" // 设置随机设备地址的静态地址
AT+BLEADDR=1 // 设置随机设备地址的私有地址
AT+BLEADDR=0 // 设置公共设备地址
```

3.4.4 AT+BLENAME: 查询/设置 Bluetooth LE 设备名称

查询命令

功能:

查询 Bluetooth LE 设备名称

命令:

```
AT+BLENAME?
```

响应:

```
+BLENAME:<device_name>
OK
```

设置命令

功能:

设置 Bluetooth LE 设备名称

命令:

```
AT+BLENAME=<device_name>
```

响应:

```
OK
```

参数

- **<device_name>**: Bluetooth LE 设备名称，最大长度：32，默认名称为“ESP-AT”。

说明

- 若 **AT+SYSTORE=1**，配置更改将保存在 NVS 区。
- 通过该命令设置设备名称后，建议你执行 **AT+BLEADVDATA** 命令将设备名称放进广播数据当中。

示例

```
AT+BLENAME="esp_demo"
```

3.4.5 AT+BLES SCANPARAM: 查询/设置 Bluetooth LE 扫描参数

查询命令

功能:

查询 Bluetooth LE 扫描参数

命令:

```
AT+BLES SCANPARAM?
```

响应:

```
+BLES SCANPARAM:<scan_type>,<own_addr_type>,<filter_policy>,<scan_interval>,<scan_
↵window>
OK
```

设置命令

功能:

设置 Bluetooth LE 扫描参数

命令:

```
AT+BLES SCANPARAM=<scan_type>,<own_addr_type>,<filter_policy>,<scan_interval>,<scan_
↵window>
```

响应:

```
OK
```

参数

- **<scan_type>**: 扫描类型
 - 0: 被动扫描 (默认)
 - 1: 主动扫描
- **<own_addr_type>**: 地址类型
 - 0: 公共地址 (默认)
 - 1: 随机地址
 - 2: RPA 公共地址
 - 3: RPA 随机地址
- **<filter_policy>**: 扫描过滤方式
 - 0: BLE_SCAN_FILTER_ALLOW_ALL (默认)
 - 1: BLE_SCAN_FILTER_ALLOW_ONLY_WLST
 - 2: BLE_SCAN_FILTER_ALLOW_UND_RPA_DIR
 - 3: BLE_SCAN_FILTER_ALLOW_WLIST_PRA_DIR
- **<scan_interval>**: 扫描间隔。本参数值应大于等于 <scan_window> 参数值。参数范围: [0x0004,0x4000]。扫描间隔是该参数乘以 0.625 毫秒,所以实际的扫描间隔范围为 [2.5,10240] 毫秒。默认值: 0x50。
- **<scan_window>**: 扫描窗口。本参数值应小于等于 <scan_interval> 参数值。参数范围: [0x0004,0x4000]。扫描窗口是该参数乘以 0.625 毫秒,所以实际的扫描窗口范围为 [2.5,10240] 毫秒。默认值: 0x50。

示例

```
AT+BLEINIT=1    // 角色：客户端
AT+BLES SCANPARAM=0,0,0,100,50
```

3.4.6 AT+BLES SCAN：使能 Bluetooth LE 扫描

设置命令

功能：

开始/停止 Bluetooth LE 扫描

命令：

```
AT+BLES SCAN=<enable>[,<duration>][,<filter_type>,<filter_param>][,<duplicate_check>]
```

响应：

```
+BLES SCAN:<addr>,<rss i>,<adv_data>,<scan_rsp_data>,<addr_type>
OK
+BLES CANDONE
```

参数

- **<enable>**:
 - 1: 开始持续扫描
 - 0: 停止持续扫描
- **[<duration>]**: 扫描持续时间, 单位: 秒。
- 若设置停止扫描, 无需设置本参数;
- 若设置开始扫描, 需设置本参数:
 - 本参数设为 0 时, 则表示开始持续扫描;
 - 本参数设为非 0 值时, 例如 AT+BLES SCAN=1, 3, 则表示扫描 3 秒后自动结束扫描, 然后返回扫描结果。
- **[<filter_type>]**: 过滤选项
 - 1: “MAC”
 - 2: “NAME”
 - 3: “UUID”
 - 4: “RSSI”
- **[<filter_param>]**: 过滤参数, 表示对方设备 MAC 地址或名称或 UUID 或 RSSI
- **[<duplicate_check>]**: 重复检查
 - 1: 进行重复检查
- **<addr>**: Bluetooth LE 地址
- **<rss i>**: 信号强度
- **<adv_data>**: 广播数据
- **<scan_rsp_data>**: 扫描响应数据
- **<addr_type>**: 广播设备地址类型

说明

- 响 应 中 的 OK 和 +BLES SCAN:<addr>,<rss i>,<adv_data>,<scan_rsp_data>,<addr_type> 在输出顺序上没有严格意义上的先后顺序。OK 可能在 +BLES SCAN:<addr>,<rss i>,<adv_data>,<scan_rsp_data>,<addr_type> 之前输出, 也有可能在这之后输出。

- 如果你想要获得扫描响应数据, 需要使用 **AT+BLESCANPARAM** 命令设置扫描方式为 active scan (AT+BLESCANPARAM=1,0,0,100,50), 并且对端设备需要设置 scan rsp data, 才能获得扫描响应数据。
- 如果根据 MAC 地址过滤, 可以支持缺省过滤, 如 AT+BLESCAN=1,3,1,"24:0A:C4"。
- 如果根据设备名称过滤, 可以支持缺省过滤, 如 AT+BLESCAN=1,3,2,"ESP"。
- UUID 过滤功能支持 16 位、32 位及 128 位格式的 UUID。参数值需以十六进制字符串形式提供。例如, 若需将过滤 UUID 设置为 "0xFF01", 可使用命令 AT+BLESCAN=1,3,3,"FF01"。
- RSSI 过滤参数的值为整数。例如, 若想设置 RSSI 过滤参数为 -50, 则命令为 AT+BLESCAN=1,3,4,-50。只会返回 RSSI 大于等于 -50 dBm 的设备。
- 如需启用重复检查功能, 须先用 **AT+BLESCAN** 命令将重复检查参数设为 1 (如果不设置此参数, 即关闭重复检查功能)。该功能需与前述过滤参数配合使用。

示例

```
AT+BLEINIT=1      // 角色：客户端
AT+BLESCAN=1      // 开始扫描
AT+BLESCAN=0      // 停止扫描
AT+BLESCAN=1,3,1,"24:0A:C4:96:E6:88" // 开始扫描，过滤类型为 MAC 地址
AT+BLESCAN=1,3,2,"ESP-AT" // 开始扫描，过滤类型为设备名称
AT+BLESCAN=1,3,3,"FF01" // 开始扫描，过滤类型为 UUID
AT+BLESCAN=1,3,4,-50 // 开始扫描，过滤类型为 RSSI
AT+BLESCAN=1,3,1,"24:0A:C4:96:E6:88",1 // 开始扫描，过滤类型为 MAC 地址，进行重复检查
```

3.4.7 AT+BLESCANRSPDATA：设置 Bluetooth LE 扫描响应

设置命令

功能：

设置 Bluetooth LE 扫描响应

命令：

```
AT+BLESCANRSPDATA=<scan_rsp_data>
```

响应：

```
OK
```

参数

- **<scan_rsp_data>**: 扫描响应数据, 为 HEX 字符串。例如, 若想设置扫描响应数据为 "0x11 0x22 0x33 0x44 0x55", 则命令为 AT+BLESCANRSPDATA="1122334455"。默认值: 空 (未设置扫描响应数据)。

示例

```
AT+BLEINIT=2      // 角色：服务器
AT+BLEADVPARAM=32,32,0,0,7,0,, // 设置广播参数
AT+BLESCANRSPDATA="1122334455" // 设置扫描响应数据
```

3.4.8 AT+BLEADVPARAM: 查询/设置 Bluetooth LE 广播参数

查询命令

功能:

查询广播参数

命令:

```
AT+BLEADVPARAM?
```

响应:

```
+BLEADVPARAM:<adv_int_min>,<adv_int_max>,<adv_type>,<own_addr_type>,<channel_map>,<adv_filter_policy>,<peer_addr_type>,<"peer_addr">,<primary_PHY>,<secondary_PHY>
OK
```

设置命令

功能:

设置广播参数

命令:

```
AT+BLEADVPARAM=<adv_int_min>,<adv_int_max>,<adv_type>,<own_addr_type>,<channel_map>
[,<adv_filter_policy>][,<peer_addr_type>,<"peer_addr">][,<primary_PHY>,<secondary_PHY>]
```

响应:

```
OK
```

参数

- **<adv_int_min>**: 最小广播间隔。参数范围: [0x0020,0x4000]。广播间隔等于该参数乘以 0.625 毫秒,所以实际的最小广播间隔范围为 [20,10240] 毫秒。本参数值应小于等于 <adv_int_max> 参数值。默认值: 0x20。
- **<adv_int_max>**: 最大广播间隔。参数范围: [0x0020,0x4000]。广播间隔等于该参数乘以 0.625 毫秒,所以实际的最大广播间隔范围为 [20,10240] 毫秒。本参数值应大于等于 <adv_int_min> 参数值。默认值: 0x40。
- **<adv_type>**:
 - 0: ADV_TYPE_IND (默认)
 - 1: ADV_TYPE_DIRECT_IND_HIGH
 - 2: ADV_TYPE_SCAN_IND
 - 3: ADV_TYPE_NONCONN_IND
 - 4: ADV_TYPE_DIRECT_IND_LOW
 - 5: ADV_TYPE_EXT_NOSCANNABLE_IND
 - 6: ADV_TYPE_EXT_CONNECTABLE_IND
 - **7: ADV_TYPE_EXT_SCANNABLE_IND**
 - 当设置广播类型为 0-4, 则使用 [AT+BLEADVDATA](#) 命令设置广播参数最多只能设置 31 字节, 如果需要设置更长的广播参数, 请调用 [AT+BLESCANRSPDATA](#) 命令来设置。
 - 当设置广播类型为 5-7, 则使用 [AT+BLEADVDATA](#) 命令设置广播参数最多只能设置 119 字节。
- **<own_addr_type>**: Bluetooth LE 地址类型
 - 0: BLE_ADDR_TYPE_PUBLIC (默认)
 - 1: BLE_ADDR_TYPE_RANDOM

- **<channel_map>**: 广播信道
 - 1: ADV_CHNL_37
 - 2: ADV_CHNL_38
 - 4: ADV_CHNL_39
 - 7: ADV_CHNL_ALL (默认)
- **[<adv_filter_policy>]**: 广播过滤器规则
 - 0: ADV_FILTER_ALLOW_SCAN_ANY_CON_ANY (默认)
 - 1: ADV_FILTER_ALLOW_SCAN_WLST_CON_ANY
 - 2: ADV_FILTER_ALLOW_SCAN_ANY_CON_WLST
 - 3: ADV_FILTER_ALLOW_SCAN_WLST_CON_WLST
- **[<peer_addr_type>]**: 对方 Bluetooth LE 地址类型
 - 0: PUBLIC (默认)
 - 1: RANDOM
- **[<" peer_addr" >]**: 对方 Bluetooth LE 地址。默认值:" 00:00:00:00:00:00"。
- **[<primary_phy>]**: 广播 primary PHY。默认值: 1M PHY。
 - 1: 1M PHY
 - 3: Coded PHY
- **[<secondary_phy>]**: 广播 secondary PHY。默认值: 1M PHY。
 - 1: 1M PHY
 - 2: 2M PHY
 - 3: Coded PHY

说明

- 如果从未设置过 peer_addr, 那么查询出来的结果会是全零。
- primary_phy 和 secondary_phy 需要一起设置, 如果不设置, 那么未设置的参数会使用默认 1M PHY。

示例

```
AT+BLEINIT=2 // 角色：服务器
AT+BLEADVPARAM=50,50,0,0,4,0,1,"12:34:45:78:66:88"
AT+BLEADVPARAM=32,32,6,0,7,0,0,"62:34:45:78:66:88",1,3
```

3.4.9 AT+BLEADVDATA: 设置 Bluetooth LE 广播数据

设置命令

功能:

设置广播数据

命令:

```
AT+BLEADVDATA=<adv_data>
```

响应:

```
OK
```

参数

- **<adv_data>**: 广播数据, 为 HEX 字符串。例如, 若想设置广播数据为 "0x11 0x22 0x33 0x44 0x55", 则命令为 AT+BLEADVDATA="1122334455"。最大长度: 119 字节。默认值: 空 (未设置广播数据)。

说明

- 如果之前已经使用命令 `AT+BLEADVDATAEX=<dev_name>,<uuid>,<manufacturer_data>,<include_power>` 设置了广播数据，则会被本命令设置的广播数据覆盖。
- 如果你想使用本命令修改设备名称，则建议在执行完该命令之后执行 `AT+BLENAME` 命令将设备名称设置为同样的名称。
- 在使用 `AT+BLEADVDATA` 命令之前，必须先通过 `AT+BLEADVPARAM` 命令设置广播参数。
- 当调用 `AT+BLEADVPARAM` 命令设置广播类型为 0-4，则使用 `AT+BLEADVDATA` 命令设置广播数据最多只能设置 31 字节，如果需要设置更长的广播数据，请调用 `AT+BLESANRSPDATA` 命令来设置。
- 当调用 `AT+BLEADVPARAM` 命令设置广播类型为 5-7，则使用 `AT+BLEADVDATA` 命令设置广播数据最多只能设置 119 字节。

示例

```
AT+BLEINIT=2    // 角色：服务器
AT+BLEADVPARAM=32,32,0,0,7,0,, // 设置广播参数
AT+BLEADVDATA="1122334455"    // 设置广播数据
```

3.4.10 AT+BLEADVDATAEX：自动设置 Bluetooth LE 广播数据

查询命令

功能：

查询广播数据的参数

命令：

```
AT+BLEADVDATAEX?
```

响应：

```
+BLEADVDATAEX:<dev_name>,<uuid>,<manufacturer_data>,<include_power>
OK
```

设置命令

功能：

设置广播数据并开始广播

命令：

```
AT+BLEADVDATAEX=<dev_name>,<uuid>,<manufacturer_data>,<include_power>
```

响应：

```
OK
```

参数

- **<dev_name>**：字符串参数，表示设备名称。例如，若想设置设备名称为 “just-test”，则命令为 `AT+BLEADVDATAEX="just-test",<uuid>,<manufacturer_data>,<include_power>`。默认值：“ESP-AT”。

- **<uuid>**: 字符串参数。例如，若想设置 UUID 为 “0xA002”，则命令为 `AT+BLEADVDATAEX=<dev_name>,"A002",<manufacturer_data>,<include_power>`。默认值：空。
- **<manufacturer_data>**: 制造商数据，为 HEX 字符串。例如，若想设置制造商数据为 “0x11 0x22 0x33 0x44 0x55”，则命令为 `AT+BLEADVDATAEX=<dev_name>,<uuid>,"1122334455",<include_power>`。默认值：空。
- **<include_power>**: 若广播数据需包含 TX 功率，本参数应该设为 1；否则，为 0。默认值：0。

说明

- 如果之前已经使用命令 `AT+BLEADVDATA=<adv_data>` 设置了广播数据，则会被本命令设置的广播数据覆盖。
- 此命令会自动将之前使用 `AT+BLEADVPARAM` 命令设置的广播类型更改为 0。
- AT 会根据用户输入的参数依次自动组包，但数据总长度最大为 31 字节，若超出该长度，将被截断。

示例

```
AT+BLEINIT=2    // 角色：服务器
AT+BLEADVDATAEX="ESP-AT","A002","0102030405",1
```

3.4.11 AT+BLEADVSTART: 开始 Bluetooth LE 广播

执行命令

功能:

开始广播

命令:

```
AT+BLEADVSTART
```

响应:

```
OK
```

说明

- 必须先使用命令 `AT+BLEADVPARAM=<adv_parameter>` 在开启广播之前设置广播参数。
- 若未使用命令 `AT+BLEADVDATA=<adv_data>` 设置广播数据，则发送全 0 数据包。
- 若之前已经使用命令 `AT+BLEADVDATA=<adv_data>` 设置过广播数据，则会被 `AT+BLEADVDATAEX=<dev_name>,<uuid>,<manufacturer_data>,<include_power>` 设置的广播数据覆盖，相反，如果先使用 `AT+BLEADVDATAEX`，则会被 `AT+BLEADVDATA` 设置的广播数据覆盖。
- 开启 Bluetooth LE 广播后，如果没有建立 Bluetooth LE 连接，那么将会一直保持广播；如果建立了连接，则会自动结束广播。

示例

```
AT+BLEINIT=2    // 角色：服务器
AT+BLEADVPARAM=50,50,0,0,7,0,,
AT+BLEADVSTART
```

3.4.12 AT+BLEADVSTOP: 停止 Bluetooth LE 广播

执行命令

功能:

停止广播

命令:

```
AT+BLEADVSTOP
```

响应:

```
OK
```

说明

- 若开始广播后，成功建立 Bluetooth LE 连接，则会自动结束 Bluetooth LE 广播，无需调用本命令。

示例

```
AT+BLEINIT=2    // 角色：服务器
AT+BLEADVPARAM=50,50,0,0,7,0,,
AT+BLEADVSTART
AT+BLEADVSTOP
```

3.4.13 AT+BLECONN: 建立 Bluetooth LE 连接

查询命令

功能:

查询 Bluetooth LE 连接

命令:

```
AT+BLECONN?
```

响应:

```
+BLECONN:<conn_index>,<remote_address>
OK
```

若未建立连接，则响应不显示 <conn_index> 和 <remote_address> 参数。

设置命令

功能:

建立 Bluetooth LE 连接

命令:

```
AT+BLECONN=<conn_index>,<remote_address>[,<addr_type>,<timeout>]
```

响应:

若建立连接成功，则提示：

```
+BLECONN:<conn_index>,<remote_address>
```

```
OK
```

若建立连接失败，则提示：

```
+BLECONN:<conn_index>,-1
```

```
ERROR
```

若是因为参数错误或者其它的一些原因导致连接失败，则提示：

```
ERROR
```

参数

- **<conn_index>**：Bluetooth LE 连接号，范围：[0,2]。
- **<remote_address>**：对方 Bluetooth LE 设备地址。
- **[<addr_type>]**：广播设备地址类型：
 - 0: 公共地址 (Public Address)
 - 1: 随机地址 (Random Address)
- **[<timeout>]**：连接超时时间，单位：秒。范围：[3,30]。

说明

- 建议在建立新连接之前，先运行 *AT+BLESCAN* 命令扫描设备，确保目标设备处于广播状态。
- 最大连接超时为 30 秒。
- 如果 Bluetooth LE server 已初始化且连接已成功建立，则可以使用此命令在对等设备 (GATTC) 中发现服务。还可以使用以下 GATTC 命令：
 - *AT+BLEGATTCPRIMSRV*
 - *AT+BLEGATTCINCLSRV*
 - *AT+BLEGATTCCHAR*
 - *AT+BLEGATTCRD*
 - *AT+BLEGATTCWR*
 - *AT+BLEGATTSIND*
- 如果 *AT+BLECONN?* 在 Bluetooth LE 未初始的情况下执行 (*AT+BLEINIT=0*)，则系统不会输出 *+BLECONN:<conn_index>,<remote_address>*。
- 支持地址解析功能。

示例

```
AT+BLEINIT=1    // 角色：客户端
AT+BLECONN=0,"24:0a:c4:09:34:23",0,10
```

3.4.14 AT+BLECONNPARAM: 查询/更新 Bluetooth LE 连接参数

查询命令

功能：

查询 Bluetooth LE 连接参数

命令：


```
AT+BLECONNPARAM?
```

响应:

```
+BLECONNPARAM:<conn_index>,<min_interval>,<max_interval>,<cur_interval>,<latency>,<timeout>
OK
```

设置命令

功能:

更新 Bluetooth LE 连接参数

命令:

```
AT+BLECONNPARAM=<conn_index>,<min_interval>,<max_interval>,<latency>,<timeout>
```

响应:

```
OK
```

若设置失败，则提示以下信息：

```
+BLECONNPARAM: <conn_index>,-1
```

参数

- **<conn_index>**: Bluetooth LE 连接号，范围：[0,2]。
- **<min_interval>**: 最小连接间隔。本参数值应小于等于 <max_interval> 参数值。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 [7.5,4000] 毫秒。
- **<max_interval>**: 最大连接间隔。本参数值应大于等于 <min_interval> 参数值。参数范围：[0x0006,0x0C80]。连接间隔等于该参数乘以 1.25 毫秒，所以实际的最大连接间隔范围为 [7.5,4000] 毫秒。
- **<cur_interval>**: 当前连接间隔。
- **<latency>**: 延迟。参数范围：[0x0000,0x01F3]。
- **<timeout>**: 超时。参数范围：[0x000A,0x0C80]。超时等于该参数乘以 10 毫秒，所以实际的超时范围为 [100,32000] 毫秒。

说明

- 本命令要求先建立连接，client 或者 server 角色都支持更新连接参数。

示例

```
AT+BLEINIT=1    // 角色：客户端
AT+BLECONN=0,"24:0a:c4:09:34:23"
AT+BLECONNPARAM=0,12,14,1,500
```

3.4.15 AT+BLEDISCONN: 断开 Bluetooth LE 连接

执行命令

功能:

断开 Bluetooth LE 连接

命令:

```
AT+BLEDISCONN=<conn_index>
```

响应:

```
OK // 收到 AT+BLEDISCONN 命令
+BLEDISCONN:<conn_index>,<remote_address> // 运行命令成功
```

参数

- **<conn_index>**: Bluetooth LE 连接号, 范围: [0,2]。
- **<remote_address>**: 对方 Bluetooth LE 设备地址。

示例

```
AT+BLEINIT=1 // 角色: 客户端
AT+BLECONN=0, "24:0a:c4:09:34:23"
AT+BLEDISCONN=0
```

3.4.16 AT+BLECFGMTU: 查询/设置 Bluetooth LE MTU 长度

查询命令

功能:

查询 MTU (maximum transmission unit, 最大传输单元) 长度

命令:

```
AT+BLECFGMTU?
```

响应:

```
+BLECFGMTU:<conn_index>,<mtu_size>
OK
```

设置命令

功能:

设置 MTU 的长度

命令:

```
AT+BLECFGMTU=<conn_index>
```

响应:

```
OK // 收到本命令
```

参数

- **<conn_index>**: Bluetooth LE 连接号, 范围: [0,2]。

说明

- 使用前需先建立 Bluetooth LE 连接。若未主动执行本命令, 连接建立后将不会主动发起 MTU 协商, 默认使用 23 字节长度。
- 客户端和服务端均可使用本命令发起 MTU 协商。本命令不接受 MTU 长度参数, 仅触发协商, 默认协商值为 203 字节。如需更高 MTU 值, 请在编译时配置: `python build.py menuconfig > Component config > Bluetooth > NimBLE Options > Preferred MTU size in octets`。

示例

```
AT+BLEINIT=1    // 角色: 客户端
AT+BLECONN=0, "24:0a:c4:09:34:23"
AT+BLECFGMTU=0
```

3.4.17 AT+BLEGATTSSRV: GATTS 发现服务

查询命令

功能:

GATTS 发现服务

命令:

```
AT+BLEGATTSSRV?
```

响应:

```
+BLEGATTSSRV:<srv_index>,<start>,<srv_uuid>,<srv_type>
OK
```

参数

- **<srv_index>**: 服务序号, 从 1 开始递增。
- **<start>**:
 - 0: 服务未开始;
 - 1: 服务已开始。
- **<srv_uuid>**: 服务的 UUID。
- **<srv_type>**: 服务的类型:
 - 0: 次要服务;
 - 1: 首要服务。

示例

```
AT+BLEINIT=2    // 角色: 服务器
AT+BLEGATTSSRV?
```

3.4.18 AT+BLEGATTCHAR: GATT 发现服务特征

查询命令

功能:

GATT 发现服务特征

命令:

```
AT+BLEGATTCHAR?
```

响应:

对于服务特征信息，响应如下：

```
+BLEGATTCHAR:"char",<srv_index>,<char_index>,<char_uuid>,<char_prop>
```

对于描述符信息，响应如下：

```
+BLEGATTCHAR:"desc",<srv_index>,<char_index>,<desc_index>,<desc_uuid>
OK
```

参数

- **<srv_index>**: 服务序号，从 1 开始递增。
- **<char_index>**: 服务特征的序号，从 1 起始递增。
- **<char_uuid>**: 服务特征的 UUID。
- **<char_prop>**: 服务特征的属性。
- **<desc_index>**: 特征描述符序号。
- **<desc_uuid>**: 特征描述符的 UUID。

示例

```
AT+BLEINIT=2    // 角色：服务器
AT+BLEGATTCHAR?
```

3.4.19 AT+BLEGATTSTNTFY: 服务器 notify 服务特征值给客户端

设置命令

功能:

服务器 notify 服务特征值给客户端

命令:

```
AT+BLEGATTSTNTFY=<conn_index>,<srv_index>,<char_index>,<length>
```

响应:

```
>
```

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

若数据传输成功，则提示：

OK

参数

- **<conn_index>**: Bluetooth LE 连接号, 范围: [0,2]。
- **<srv_index>**: 服务序号, 可运行 [AT+BLEGATTSCCHAR?](#) 查询。
- **<char_index>**: 服务特征的序号, 可运行 [AT+BLEGATTSCCHAR?](#) 查询。
- **<length>**: 数据长度, 最大长度: (*MTU* - 3)。

说明

- 数据单次发送的最大长度为 (*MTU* - 3) 字节。若设置的长度超过此值, 底层会自动分包传输。

示例

```
AT+BLEINIT=2          // 角色: 服务器
AT+BLEADVPARAM=50,50,0,0,7,0,,
AT+BLEADVSTART        // 开始广播, 当 client 连接后, 必须配置接收 notify
AT+BLEGATTSCCHAR?     // 查询允许 notify 客户端的特征
// 例如, 使用 3 号服务的 6 号特征 notify 长度为 4 字节的数据, 使用如下命令:
AT+BLEGATTSENTFY=0,3,6,4
// 提示 ">" 符号后, 输入 4 字节的数据, 如 "1234", 然后数据自动传输
```

3.4.20 AT+BLEGATTSEND: 服务器 indicate 服务特征值给客户端

设置命令

功能:

服务器 indicate 服务特征值给客户端

命令:

```
AT+BLEGATTSEND=<conn_index>,<srv_index>,<char_index>,<length>
```

响应:

>

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

若数据传输成功, 则提示:

OK

参数

- **<conn_index>**: Bluetooth LE 连接号, 范围: [0,2]。
- **<srv_index>**: 服务序号, 可运行 [AT+BLEGATTSCCHAR?](#) 查询。
- **<char_index>**: 服务特征的序号, 可运行 [AT+BLEGATTSCCHAR?](#) 查询。
- **<length>**: 数据长度, 最大长度: (*MTU* - 3)。

示例

```

AT+BLEINIT=2      // 角色：服务器
AT+BLEADVPARAM=50,50,0,0,7,0,,
AT+BLEADVSTART    // 开始广播，当 client 连接后，必须配置接收 indication
AT+BLEGATTCHAR?   // 查询客户端可以接收 indication 的特征
// 例如，使用 3 号服务的 7 号特征 indicate 长度为 4 字节的数据，命令如下：
AT+BLEGATTSSIND=0,3,7,4
// 提示 ">" 符号后，输入 4 字节的数据，如 "1234"，然后数据自动传输

```

3.4.21 AT+BLEGATTSSSETATTR: GATTS 设置服务特征值

设置命令

功能：

GATTS 设置服务特征值或描述符值

命令：

```
AT+BLEGATTSSSETATTR=<srv_index>,<char_index>,<length>
```

响应：

```
>
```

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

若数据传输成功，则提示：

```
OK
```

参数

- <srv_index>：服务序号，可运行 [AT+BLEGATTCHAR?](#) 查询。
- <char_index>：服务特征的序号，可运行 [AT+BLEGATTCHAR?](#) 查询。
- 若填写，则设置描述符的值；
- 若未填写，则设置特征值。
- <length>：数据长度。

说明

- 如果 <length> 参数值大于支持的最大长度，则设置会失败。关于 service table，请见 [gatts_data.csv](#)。
- 无法设置特征值描述符的值。

示例

```

AT+BLEINIT=2    // 角色：服务器
AT+BLEGATTCHAR?
// 例如，向 1 号服务的 1 号特征写入长度为 1 字节的数据，命令如下：
AT+BLEGATTSSSETATTR=1,1,1
// 提示 ">" 符号后，输入 1 字节的数据即可，例如 "8"，然后设置开始

```

3.4.22 AT+BLEGATTCPRIMSRV: GATTC 发现基本服务

查询命令

功能:

GATTC (Generic Attributes Client) 发现基本服务

命令:

```
AT+BLEGATTCPRIMSRV=<conn_index>
```

响应:

```
+BLEGATTCPRIMSRV:<conn_index>,<srv_index>,<srv_uuid>,<srv_type>  
OK
```

参数

- **<conn_index>**: Bluetooth LE 连接号, 范围: [0,2]。
- **<srv_index>**: 服务序号, 从 1 开始递增。
- **<srv_uuid>**: 服务的 UUID。
- **<srv_type>**: 服务的类型:
 - 0: 次要服务;
 - 1: 首要服务。

说明

- 使用本命令, 需要先建立 Bluetooth LE 连接。

示例

```
AT+BLEINIT=1 // 角色: 客户端  
AT+BLECONN=0, "24:12:5f:9d:91:98"  
AT+BLEGATTCPRIMSRV=0
```

3.4.23 AT+BLEGATTCINCLSRV: GATTC 发现包含的服务

设置命令

功能:

GATTC 发现包含服务

命令:

```
AT+BLEGATTCINCLSRV=<conn_index>,<srv_index>
```

响应:

```
+BLEGATTCINCLSRV:<conn_index>,<srv_index>,<srv_uuid>,<srv_type>,<included_srv_uuid>  
↔,<included_srv_type>  
OK
```

参数

- **<conn_index>**: Bluetooth LE 连接号, 范围: [0,2]。
- **<srv_index>**: 服务序号, 可运行 `AT+BLEGATTCPRIMSRV=<conn_index>` 查询。
- **<srv_uuid>**: 服务的 UUID。
- **<srv_type>**: 服务的类型:
 - 0: 次要服务;
 - 1: 首要服务。
- **<included_srv_uuid>**: 包含服务的 UUID。
- **<included_srv_type>**: 包含服务的类型:
 - 0: 次要服务;
 - 1: 首要服务。

说明

- 使用本命令, 需要先建立 Bluetooth LE 连接。

示例

```
AT+BLEINIT=1 // 角色: 客户端
AT+BLECONN=0, "24:12:5f:9d:91:98"
AT+BLEGATTCPRIMSRV=0
AT+BLEGATTTCINCLSRV=0,1 // 根据前一条命令的查询结果, 指定 index 查询
```

3.4.24 AT+BLEGATTCHAR: GATT 发现服务特征

设置命令

功能:

GATT 发现服务特征

命令:

```
AT+BLEGATTCHAR=<conn_index>,<srv_index>
```

响应:

对于服务特征信息, 响应如下:

```
+BLEGATTCHAR:"char",<conn_index>,<srv_index>,<char_index>,<char_uuid>,<char_prop>
```

对于描述符信息, 响应如下:

```
+BLEGATTCHAR:"desc",<conn_index>,<srv_index>,<char_index>,<desc_index>,<desc_uuid>
OK
```

参数

- **<conn_index>**: Bluetooth LE 连接号, 范围: [0,2]。
- **<srv_index>**: 服务序号, 可运行 `AT+BLEGATTCPRIMSRV=<conn_index>` 查询。
- **<char_index>**: 服务特征的序号, 从 1 开始递增。
- **<char_uuid>**: 服务特征的 UUID。
- **<char_prop>**: 服务特征的属性。
- **<desc_index>**: 特征描述符序号。
- **<desc_uuid>**: 特征描述符的 UUID。

说明

- 使用本命令，需要先建立 Bluetooth LE 连接。

示例

```
AT+BLEINIT=1 // 角色：客户端
AT+BLECONN=0,"24:12:5f:9d:91:98"
AT+BLEGATTCPRIMSRV=0
AT+BLEGATTCCCHAR=0,1 // 根据前一条命令的查询结果，指定 index 查询
```

3.4.25 AT+BLEGATTCD: GATT 读取服务特征值

设置命令

功能:

GATT 读取服务特征值或描述符值

命令:

```
AT+BLEGATTCD=<conn_index>,<srv_index>,<char_index>[,<desc_index>]
```

响应:

```
+BLEGATTCD:<conn_index>,<len>,<value>
OK
```

参数

- **<conn_index>**: Bluetooth LE 连接号，范围：[0,2]。
- **<srv_index>**: 服务序号，可运行 [AT+BLEGATTCPRIMSRV=<conn_index>](#) 查询。
- **<char_index>**: 服务特征序号，可运行 [AT+BLEGATTCCCHAR=<conn_index>,<srv_index>](#) 查询。
- **[<desc_index>]**: 特征描述符序号：
 - 若设置，读取目标描述符的值；
 - 若未设置，读取目标特征的值。
- **<len>**: 数据长度。
- **<value>**: <char_value> 或者 <desc_value>。
- **<char_value>**: 服务特征值，字符串格式，运行 [AT+BLEGATTCD=<conn_index>,<srv_index>,<char_index>](#) 读取。例如，若响应为 +BLEGATTCD:0,1,0，则表示数据长度为 1，内容为“0”。
- **<desc_value>**: 服务特征描述符的值，字符串格式，运行 [AT+BLEGATTCD=<conn_index>,<srv_index>,<char_index>,<desc_index>](#) 读取。例如，若响应为 +BLEGATTCD:0,4,0123，则表示数据长度为 4，内容为“0123”。

说明

- 使用本命令，需要先建立 Bluetooth LE 连接。
- 若目标服务特征不支持读操作，则返回“ERROR”。

示例

```

AT+BLEINIT=1    // 角色：客户端
AT+BLECONN=0,"24:12:5f:9d:91:98"
AT+BLEGATTCPRIMSRV=0
AT+BLEGATTCCHAR=0,3 // 根据前一条命令的查询结果，指定 index 查询
// 例如，读取第 3 号服务的第 2 号特征的第 1 号描述符信息，命令如下：
AT+BLEGATTCRD=0,3,2,1

```

3.4.26 AT+BLEGATTCWR: GATTC 写服务特征值

设置命令

功能：

GATTC 写服务特征值或描述符值

命令：

```
AT+BLEGATTCWR=<conn_index>,<srv_index>,<char_index>[,<desc_index>],<length>
```

响应：

```
>
```

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

若数据传输成功，则提示：

```
OK
```

参数

- **<conn_index>**：Bluetooth LE 连接号，范围：[0,2]。
- **<srv_index>**：服务序号，可运行 [AT+BLEGATTCPRIMSRV=<conn_index>](#) 查询。
- **<char_index>**：服务特征序号，可运行 [AT+BLEGATTCCHAR=<conn_index>,<srv_index>](#) 查询。
- **[<desc_index>]**：特征描述符序号：
 - 若设置，则写目标描述符的值；
 - 若未设置，则写目标特征的值。
- **<length>**：数据长度。该参数的取值范围受 [gatts_data.csv](#) 中 val_max_len 参数影响。

说明

- 使用本命令，需要先建立 Bluetooth LE 连接。
- 若目标服务特征不支持写操作，则返回“ERROR”。

示例

```

AT+BLEINIT=1    // 角色：客户端
AT+BLECONN=0,"24:12:5f:9d:91:98"
AT+BLEGATTCPRIMSRV=0
AT+BLEGATTCCHAR=0,3 // 根据前一条命令的查询结果，指定 index 查询
// 例如，向第 3 号服务的第 4 号特征，写入长度为 6 字节的数据，命令如下：
AT+BLEGATTCWR=0,3,4,,6
// 提示 ">" 符号后，输入 6 字节的数据即可，如 "123456"，然后开始写入

```

3.4.27 AT+BLESPPCFG: 查询/设置 Bluetooth LE SPP 参数

查询命令

功能:

查询 Bluetooth LE SPP (Serial Port Profile) 参数

命令:

```
AT+BLESPPCFG?
```

响应:

```
+BLESPPCFG:<tx_service_index>,<tx_char_index>,<rx_service_index>,<rx_char_index>,  
↪<auto_conn>  
OK
```

设置命令

功能:

设置或重置 Bluetooth LE SPP 参数

命令:

```
AT+BLESPPCFG=<cfg_enable>[,<tx_service_index>,<tx_char_index>,<rx_service_index>,  
↪<rx_char_index>][,<auto_conn>]
```

响应:

```
OK
```

参数

- **<cfg_enable>**:
 - 0: 重置所有 SPP 参数, 后面参数无需填写;
 - 1: 后面参数需要填写。
- **<tx_service_index>**: tx 服务序号, 可运行 [AT+BLEGATTCPRIMSRV=<conn_index>](#) 和 [AT+BLEGATTSSRV?](#) 查询。
- **<tx_char_index>**: tx 服务特征序号, 可运行 [AT+BLEGATTCCCHAR=<conn_index>,<srv_index>](#) 和 [AT+BLEGATTSCCHAR?](#) 查询。
- **<rx_service_index>**: rx 服务序号, 可运行 [AT+BLEGATTCPRIMSRV=<conn_index>](#) 和 [AT+BLEGATTSSRV?](#) 查询。
- **<rx_char_index>**: rx 服务特征序号, 可运行 [AT+BLEGATTCCCHAR=<conn_index>,<srv_index>](#) 和 [AT+BLEGATTSCCHAR?](#) 查询。
- **<auto_conn>**: 自动重连标志位, 默认情况下, 自动重连功能被使能。
 - 0: 禁止 Bluetooth LE 透传自动重连功能。
 - 1: 使能 Bluetooth LE 透传自动重连功能。

说明

- 对于 Bluetooth LE 客户端, tx 服务特征属性必须是 write with response 或 write without response, rx 服务特征属性必须是 indicate 或 notify。
- 对于 Bluetooth LE 服务器, tx 服务特征属性必须是 indicate 或 notify, rx 服务特征属性必须是 write with response 或 write without response。

- 禁用了自动重连功能后，如果连接断开，会提示有断开连接信息提示 (依赖于 AT+SYMSMSG)，需要重新发送连接的命令；使能的情况下，连接断开后，会自动重连，MCU 侧将感知不到连接的断开，如果对端的 MAC 发生了改变，则无法连接成功。

示例

```
AT+BLESPPCFG=0           // 重置 Bluetooth LE SPP 参数
AT+BLESPPCFG=1,3,5,3,7   // 设置 Bluetooth LE SPP 参数
AT+BLESPPCFG?             // 查询 Bluetooth LE SPP 参数
```

3.4.28 AT+BLESPP: 进入 Bluetooth LE SPP 模式

执行命令

功能:

进入 Bluetooth LE SPP 模式

命令:

```
AT+BLESPP
```

响应:

```
OK
>
```

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。

若 Bluetooth LE SPP 状态错误 (对端在 Bluetooth LE 连接建立后未使能 Notifications)，则返回：

```
ERROR
```

说明

- 在 SPP 传输中，若未设置 **AT+SYMSMSG** Bit0 为 1，则 AT 不会提示任何退出 SPP 透传模式的信息。
- 在 SPP 传输中，若未设置 **AT+SYMSMSG** Bit2 为 1，则 AT 不会提示任何连接状态变更的信息。
- 当系统收到只含有 +++ 的包时，设备返回到普通命令模式，请至少等待一秒再发送下一个 AT 命令。

示例

```
AT+BLESPP // 进入 Bluetooth LE SPP 模式
```

3.4.29 AT+BLESECPARAM: 查询/设置 Bluetooth LE 加密参数

查询命令

功能:

查询 Bluetooth LE SMP 加密参数

命令:

```
AT+BLESECPARAM?
```

响应:

```
+BLESECPARAM:<auth_req>,<iocap>,<enc_key_size>,<init_key>,<rsp_key>,<auth_option>
OK
```

设置命令**功能:**

设置 Bluetooth LE SMP 加密参数

命令:

```
AT+BLESECPARAM=<auth_req>,<iocap>,<enc_key_size>,<init_key>,<rsp_key>[,<auth_
↪option>]
```

响应:

```
OK
```

参数

- **<auth_req>**: 认证请求。
 - 0: NO_BOND (默认)
 - 1: BOND
 - 4: MITM
 - 8: SC_ONLY
 - 9: SC_BOND
 - 12: SC_MITM
 - 13: SC_MITM_BOND
- **<iocap>**: 输入输出能力。
 - 0: DisplayOnly (默认)
 - 1: DisplayYesNo
 - 2: KeyboardOnly
 - 3: NoInputNoOutput
 - 4: Keyboard display
- **<enc_key_size>**: 加密密钥长度。参数范围: [7,16]。单位: 字节。默认值: 0 (未设置; 将在配对过程中协商)。
- **<init_key>**: 多个比特位组成的初始密钥。默认值: 0。
- **<rsp_key>**: 多个比特位组成的响应密钥。默认值: 0。
- **<auth_option>**: 安全认证选项:
 - 0: 自动选择安全等级 (默认);
 - 1: 如果无法满足之前设定的安全等级, 则会断开连接。

说明

- **<init_key>** 和 **<rsp_key>** 参数的比特位组合模式如下:
 - Bit0: 用于交换初始密钥和响应密钥的加密密钥;
 - Bit1: 用于交换初始密钥和响应密钥的 IRK 密钥;
 - Bit2: 用于交换初始密钥和响应密钥的 CSRK 密钥;
 - Bit3: 用于交换初始密钥和响应密钥的 link 密钥 (仅用于 Bluetooth LE 和 BR/EDR 共存模式)。

示例

```
AT+BLESECPARAM=1,4,16,3,3,0
```

3.4.30 AT+BLEENC: 发起 Bluetooth LE 加密请求

设置命令

功能:

发起配对请求

命令:

```
AT+BLEENC=<conn_index>,<sec_act>
```

响应:

```
OK
```

参数

- **<conn_index>**: Bluetooth LE 连接号, 范围: [0,2]。
- **<sec_act>**:
 - 0: SEC_NONE;
 - 1: SEC_ENCRYPT;
 - 2: SEC_ENCRYPT_NO_MITM;
 - 3: SEC_ENCRYPT_MITM。

说明

- 使用本命令前, 请先设置安全参数、建立与对方设备的连接。

示例

```
AT+RESTORE
AT+BLEINIT=2
AT+BLEADDR?
AT+BLESECPARAM=1,0,16,3,3
AT+BLESETKEY=123456
AT+BLEADVPARAM=50,50,0,0,7,0,,
AT+BLEADVSTART
// 使用 Bluetooth LE 调试 app 作为 client 与 ESP32-C6 设备建立 Bluetooth LE 连接
AT+BLEENC=0,3
```

3.4.31 AT+BLEKEYREPLY: 给对方设备回复密钥

设置命令

功能:

回复配对密钥

命令:

```
AT+BLEKEYREPLY=<conn_index>,<key>
```

响应:

```
OK
```

参数

- **<conn_index>**: Bluetooth LE 连接号, 范围: [0,2]。
- **<key>**: 配对密钥。

示例

```
AT+BLEKEYREPLY=0,649784
```

3.4.32 AT+BLECONFREPLY: 给对方设备回复确认结果 (传统连接阶段)

设置命令

功能:

回复配对结果

命令:

```
AT+BLECONFREPLY=<conn_index>,<confirm>
```

响应:

```
OK
```

参数

- **<conn_index>**: Bluetooth LE 连接号, 范围: [0,2]。
- **<confirm>**:
 - 0: 否
 - 1: 是

示例

```
AT+BLECONFREPLY=0,1
```

3.4.33 AT+BLENCDEV: 查询绑定的 Bluetooth LE 加密设备列表

查询命令

功能:

查询绑定的 Bluetooth LE 加密设备列表

命令:

```
AT+BLEENCDEV?
```

响应:

```
+BLEENCDEV:<enc_dev_index>,<mac_address>
OK
```

参数

- **<enc_dev_index>**: 已绑定设备的连接号。该参数不一定等于命令 [AT+BLECONN](#) 查询出的 Bluetooth LE 连接列表中的 `conn_index` 参数。范围: [0,14]。
- **<mac_address>**: MAC 地址。

说明

- ESP-AT 最多允许绑定 15 个设备。如果绑定的设备数量超过 15 个, 那么新绑定的设备信息会根据绑定顺序从 0 到 14 号依次覆盖之前的设备信息。

示例

```
AT+BLEENCDEV?
```

3.4.34 AT+BLEENCCLEAR: 清除 Bluetooth LE 加密设备列表

设置命令

功能:

从安全数据库列表中删除某一连接号的设备

命令:

```
AT+BLEENCCLEAR=<enc_dev_index>
```

响应:

```
OK
```

执行命令

功能:

删除安全数据库所有设备

命令:

```
AT+BLEENCCLEAR
```

响应:

```
OK
```


参数

- **<enc_dev_index>**: 已绑定设备的连接号。

示例

```
AT+BLEENCCLEAR
```

3.4.35 AT+BLESETKEY: 设置 Bluetooth LE 静态配对密钥

查询命令

功能:

查询 Bluetooth LE 静态配对密钥, 若未设置, 则 AT 返回 -1

命令:

```
AT+BLESETKEY?
```

响应:

```
+BLESETKEY:<static_key>  
OK
```

设置命令

功能:

为所有 Bluetooth LE 连接设置一个 Bluetooth LE 静态配对密钥

命令:

```
AT+BLESETKEY=<static_key>
```

响应:

```
OK
```

参数

- **<static_key>**: Bluetooth LE 静态配对密钥。

示例

```
AT+BLESETKEY=123456
```

3.4.36 AT+BLUFI: 开启或关闭 BluFi

查询命令

功能:

查询 BluFi 状态

命令：

```
AT+BLUFI?
```

响应：

若 BluFi 未开启，则返回：

```
+BLUFI:0  
OK
```

若 BluFi 已开启，则返回：

```
+BLUFI:1  
OK
```

设置命令**功能：**

开启或关闭 BluFi

命令：

```
AT+BLUFI=<option>[,<auth floor>]
```

响应：

```
OK
```

参数

- **<option>:**
 - 0: 关闭 BluFi;
 - 1: 开启 BluFi。
- **<auth floor>:** Wi-Fi 认证模式阈值，ESP-AT 不会连接到认证模式低于此阈值的 AP：
 - 0: OPEN（默认）
 - 1: WEP
 - 2: WPA_PSK
 - 3: WPA2_PSK
 - 4: WPA_WPA2_PSK
 - 5: WPA2_ENTERPRISE
 - 6: WPA3_PSK
 - 7: WPA2_WPA3_PSK
 - 8: WAPI_PSK
 - 9: OWE
 - 10: WPA3_ENT_192
 - 11: WPA3_EXT_PSK
 - 12: WPA3_EXT_PSK_MIXED_MODE
 - 13: DPP
 - 14: WPA3_ENTERPRISE
 - 15: WPA2_WPA3_ENTERPRISE

说明

- 你只能在 Bluetooth LE 未初始化情况下开启或关闭 BluFi (*AT+BLEINIT=0*)。

- BluFi 配网过程中，ESP32-C6 收到手机端发送的 SSID 和密码时，会分别输出 `+SSID:<ssid>` 和 `+PASSWORD:<password>`。
- 为获得更好的性能，建议在使用 BluFi 功能前，先发送 `AT+CWMODE=0/1` 命令禁用 SoftAP。如果你想了解更多细节，请阅读 [RF 共存](#) 文档。
- BluFi 配网后请发送 `AT+BLUFI=0` 命令关闭 BluFi，以释放资源。

示例

```
AT+BLUFI=1
```

3.4.37 AT+BLUFINAME: 查询/设置 BluFi 设备名称

查询命令

功能:

查询 BluFi 名称

命令:

```
AT+BLUFINAME?
```

响应:

```
+BLUFINAME:<device_name>
OK
```

设置命令

功能:

设置 BluFi 设备名称

命令:

```
AT+BLUFINAME=<device_name>
```

响应:

```
OK
```

参数

- **<device_name>**: BluFi 设备名称。

说明

- 如需设置 BluFi 设备名称，请在运行 `AT+BLUFI=1` 命令前设置，否则将使用默认名称 `BLUFI_DEVICE`。
- BluFi 设备名称最大长度为 26 字节。
- Blufi APP 可以在应用商店中下载。

示例

```
AT+BLUFINAME="BLUFI_DEV"  
AT+BLUFINAME?
```

3.4.38 AT+BLUFISEND：发送 BluFi 用户自定义数据

设置命令

功能：

发送 BluFi 用户自定义数据给手机端

命令：

```
AT+BLUFISEND=<length>
```

响应：

```
>
```

符号 > 表示 AT 准备好接收串口数据，此时你可以输入数据，当数据长度达到参数 <length> 的值时，开始传输数据。

若数据传输成功，则提示：

```
OK
```

参数

- **<length>**：数据长度，单位：字节。

说明

- 自定义数据的长度不能超过 600 字节。
- 如果 ESP 收到手机发来的用户自定义数据，那么会以 +BLUFIDATA:<len>,<data> 格式打印。

示例

```
AT+BLUFISEND=4  
// 提示 ">" 符号后，输入 4 字节的数据即可，如 "1234"，然后数据会被自动发送给手机
```

3.4.39 AT+BLEPERIODICDATA：设置 Bluetooth LE 周期性广播数据

设置命令

功能：

设置周期性广播数据。

命令：

```
AT+BLEPERIODICDATA=<periodic_data>
```

响应：

```
OK
```

参数

- **<periodic_data>**: 周期性广播数据, 为 16 进制字符串。例如, 若想设置广播数据为 “0x11 0x22 0x33 0x44 0x55”, 则命令为 `AT+BLEPERIODICDATA="1122334455"`。默认值: 空 (未设置周期性广播数据)。

示例

```
AT+BLEINIT=2
AT+BLEPERIODICDATA="1122334455"
```

3.4.40 AT+BLEPERIODICSTART: 开启周期性广播

执行命令

功能:

开启周期性广播。

命令:

```
AT+BLEPERIODICSTART
```

响应:

```
OK
```

说明

- 在开始周期性广播之前, 需要先开启扩展广播, 扩展广播类型为 `ADV_TYPE_EXT_NOSCANNABLE_IND`。

示例

```
AT+BLEINIT=2
AT+BLEPERIODICDATA="1122334455" // 设置周期性广播数据
AT+BLEADVPARAM=32,32,5,0,7,0 // 设置扩展广播参数
AT+BLEADVSTART // 开启扩展广播
AT+BLEPERIODICSTART // 开启周期性广播
```

3.4.41 AT+BLEPERIODICSTOP: 停止周期性广播同步

执行命令

功能:

停止周期性广播

命令:

```
AT+BLEPERIODICSTOP
```

响应:

```
OK
```

示例

```
AT+BLEPERIODICSTOP // 停止周期性广播
```

3.4.42 AT+BLESYNCSTART: 开启同步周期性广播

设置命令

功能:

与正在进行周期性广播的设备同步。

命令:

```
AT+BLESYNCSTART=<target_address>
```

响应:

```
+BLESYNC:<addr>,<rssi>,<periodic_adv_data>  
OK
```

参数

- **<target_address>**: 目标设备地址, 即正在进行周期性广播的对端设备地址。
- **<addr>**: 设备地址。
- **<rssi>**: 信号强度。
- **<periodic_adv_data>**: 周期性广播数据。

说明

- 在开启周期性广播同步之前, 需要保持 Bluetooth LE 扫描功能持续进行。

示例

```
AT+BLEINIT=1  
AT+BLESCAN=1 // 开始扫描  
AT+BLESYNCSTART="24:0a:c4:09:34:23" // 开始周期性广播同步
```

3.4.43 AT+BLESYNCSTOP: 停止周期性广播同步

执行命令

功能:

停止周期性广播同步功能。

命令:

```
AT+BLESYNCTOP
```

响应:

```
OK
```

说明

- 如果客户将 BLE 扫描功能关闭，那么周期性广播同步功能也会被自动停止。

示例

```
AT+BLEINIT=1
AT+BLESCAN=1
AT+BLESYNCTOP="24:0a:c4:09:34:23"
AT+BLESYNCTOP
```

3.4.44 AT+BLERADPHY: 查询当前连接使用的 PHY

设置命令

功能:

查询当前连接使用的 PHY。

命令:

```
AT+BLERADPHY=<conn_index>
```

响应:

如果查询成功，返回:

```
+BLERADPHY:<device_addr>,<tx_phy>,<rx_phy>
OK
```

如果查询失败，返回:

```
+BLERADPHY:-1
OK
```

参数

- **<device_addr>**: 对端设备地址
- **<tx_phy>**:
 - 1: 1M PHY.
 - 2: 2M PHY.
 - 3: Coded PHY.
- **<rx_phy>**:
 - 1: 1M PHY.
 - 2: 2M PHY.
 - 3: Coded PHY.

示例

```
AT+BLEINIT=1
AT+BLECONN=0,"24:0a:c4:09:34:23"
AT+BLERADPHY=0 // 查询当前连接的 PHY
```

3.4.45 AT+BLESETPHY: 设置当前连接的 PHY

设置命令

功能:

设置当前连接的 PHY。

命令:

```
AT+BLESETPHY=<conn_index>,<tx_rx_phy>
```

响应:

如果查询成功, 返回:

```
+BLESETPHY:<device_addr>,<tx_phy>,<rx_phy>
OK
```

如果查询失败, 返回:

```
+BLESETPHY:-1
OK
```

参数

- **<device_addr>**: 对端设备地址
- **<tx_rx_phy>**:
 - 1: 1M PHY
 - 2: 2M PHY
 - 3: Coded PHY

示例

```
AT+BLEINIT=1 // 角色: 客户端
AT+BLECONN=0,"24:0a:c4:09:34:23"
AT+BLERADPHY=0
```

3.4.46 AT+BLEOTANAME: 查询/设置 Bluetooth LE OTA 设备名称

查询命令

功能:

查询 Bluetooth LE OTA 设备名称

命令:

```
AT+BLEOTANAME?
```

响应:


```
+BLEOTANAME:<device_name>
OK
```

设置命令

功能:

设置 Bluetooth LE OTA 设备名称

命令:

```
AT+BLEOTANAME=<device_name>
```

响应:

```
OK
```

参数

- **<device_name>**: Bluetooth LE OTA 设备名称, 最大长度: 29 字节, 默认名称为 “ESP-C919”。

说明

- 如需设置 Bluetooth LE OTA 设备名称, 请在运行 *AT+BLEOTA=1* 命令前设置, 否则将使用默认名称 ESP-C919。

示例

```
AT+BLEOTANAME="NAME"
AT+BLEOTANAME?
```

3.4.47 AT+BLEOTA: 初始化或反初始化 Bluetooth LE OTA

查询命令

功能:

查询 Bluetooth LE OTA 初始化状态

命令:

```
AT+BLEOTA?
```

响应:

若 Bluetooth LE OTA 未初始化, 则返回:

```
+BLEOTA:0
OK
```

若 Bluetooth LE OTA 已初始化, 则返回:

```
+BLEOTA:1
OK
```

设置命令

功能:

初始化或反初始化 Bluetooth LE OTA

命令:

```
AT+BLEOTA=<init>
```

响应:

```
OK
```

参数

- **<init>**:
 - 0: 反初始化 Bluetooth LE OTA;
 - 1: 初始化 Bluetooth LE OTA。

说明

- 默认 AT 固件未使能本命令。如需使用, 请参考[如何实现 Bluetooth LE OTA 升级](#) 开启配置 (并使能时同步将 MTU 建议设置为 512) 后重新编译。
- 如需设置 Bluetooth LE OTA 设备名称, 请在运行本命令初始化前执行 `AT+BLEOTANAME` 命令。
- 初始化成功后, 设备会自动开启广播, 等待对端 APP 连接, 然后从对端 APP 接收 OTA 固件。
- 固件传输完成并校验通过后, 设备会自动重启。
- 请参考[如何实现 Bluetooth LE OTA 升级](#) 获取 GATT 服务、对端 APP 操作顺序和数据包格式说明。

示例

```
AT+BLEOTANAME="NAME"
AT+BLEOTA=1
```

3.5 MQTT AT 命令集

- [介绍](#)
- `AT+MQTTUSERCFG`: 设置 MQTT 用户属性
- `AT+MQTTLONGCLIENTID`: 设置 MQTT 客户端 ID
- `AT+MQTTLONGUSERNAME`: 设置 MQTT 登陆用户名
- `AT+MQTTLONGPASSWORD`: 设置 MQTT 登陆密码
- `AT+MQTTCONNCFG`: 设置 MQTT 连接属性
- `AT+MQTTALPN`: 设置 MQTT 应用层协议协商 (ALPN)
- `AT+MQTTSDNI`: 设置 MQTT 服务器名称指示 (SNI)
- `AT+MQTTCONN`: 连接 MQTT Broker
- `AT+MQTTPUB`: 发布 MQTT 消息 (字符串)
- `AT+MQTTPUBRAW`: 发布长 MQTT 消息
- `AT+MQTTSUB`: 订阅 MQTT Topic
- `AT+MQTTUNSUB`: 取消订阅 MQTT Topic
- `AT+MQTTCLEAN`: 断开 MQTT 连接
- [MQTT AT 错误码](#)
- [MQTT AT 说明](#)

3.5.1 介绍

重要:

- 当前，ESP32-C6 系列 AT 固件支持 [MQTT 3.1.1 版本](#)。
- 默认的 AT 固件支持此页面下的所有 AT 命令。如果你不需要 ESP32-C6 支持 MQTT 命令，请自行[编译 ESP-AT 工程](#)，在第五步配置工程里选择：
 - 禁用 Component config>AT>AT MQTT command support
- MQTT 连接本地 Broker 示例见[MQTT AT 示例](#)
- MQTT 连接 AWS IoT 示例见[MQTT 云连接 AT 示例](#)

3.5.2 AT+MQTTUSERCFG: 设置 MQTT 用户属性

设置命令

功能:

配置 MQTT 用户属性

命令:

```
AT+MQTTUSERCFG=<LinkID>,<scheme>,<"client_id">,<"username">,<"password">,<cert_key_ID>,<CA_ID>,<"path">
```

响应:

OK

参数

- **<LinkID>**: 当前仅支持 link ID 0。
- **<scheme>**:
 - 1: MQTT over TCP;
 - 2: MQTT over TLS (不校证书);
 - 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);
 - 6: MQTT over WebSocket (基于 TCP);
 - 7: MQTT over WebSocket Secure (基于 TLS, 不校证书);
 - 8: MQTT over WebSocket Secure (基于 TLS, 校验 server 证书);
 - 9: MQTT over WebSocket Secure (基于 TLS, 提供 client 证书);
 - 10: MQTT over WebSocket Secure (基于 TLS, 校验 server 证书并且提供 client 证书)。
- **<"client_id">**: MQTT 客户端 ID, 最大长度: 256 字节。
- **<"username">**: 用户名, 用于登陆 MQTT broker, 最大长度: 64 字节。
- **<"password">**: 密码, 用于登陆 MQTT broker, 最大长度: 64 字节。
- **<cert_key_ID>**: 证书 ID, 目前 ESP-AT 仅支持一套 cert 证书, 参数为 0。
- **<CA_ID>**: CA ID, 目前 ESP-AT 仅支持一套 CA 证书, 参数为 0。
- **<"path">**: 资源路径, 最大长度: 32 字节。

说明

- 每条 AT 命令的总长度不能超过 256 字节。
- 如果你想使用自己的证书, 运行时请使用 [AT+SYSMFG](#) 命令更新 MQTT 证书 (具体步骤请参考 [AT+SYSMFG 命令示例](#), 证书配置方法与 SSL 证书相同)。如果你想预烧录自己的证书, 请参考[如何更新 PKI 配置](#)。

- 如果 <scheme> 配置为 3、5、8、10，为了校验服务器的证书有效期，请在发送 *AT+MQTTCONN* 命令前确保 ESP32-C6 已获取到当前时间（你可以发送 *AT+CIPSNTPCFG* 命令来配置 SNTP，获取当前时间，发送 *AT+CIPSNPTIME?* 命令查询当前时间）。
- 关于 TLS 协议版本配置：请参考 [修改 TLS 协议版本说明](#)。

3.5.3 AT+MQTTLONGCLIENTID: 设置 MQTT 客户端 ID

设置命令

功能：

设置 MQTT 客户端 ID

命令：

```
AT+MQTTLONGCLIENTID=<LinkID>,<length>
```

响应：

```
OK
>
```

上述响应表示 AT 已准备好接收 MQTT 客户端 ID，此时你可以输入客户端 ID，当 AT 接收到的客户端 ID 长度达到 <length> 后，返回：

```
OK
```

参数

- <LinkID>：当前仅支持 link ID 0。
- <length>：MQTT 客户端 ID 长度。范围：[1,1024]。

说明

- *AT+MQTTUSERCFG* 命令也可以设置 MQTT 客户端 ID，二者之间的差别包括：
 - *AT+MQTTLONGCLIENTID* 命令可以用来设置相对较长的客户端 ID，因为 *AT+MQTTUSERCFG* 命令的长度受限；
 - 应在设置 *AT+MQTTUSERCFG* 后再使用 *AT+MQTTLONGCLIENTID*。

3.5.4 AT+MQTTLONGUSERNAME: 设置 MQTT 登陆用户名

设置命令

功能：

设置 MQTT 用户名

命令：

```
AT+MQTTLONGUSERNAME=<LinkID>,<length>
```

响应：

```
OK
>
```

上述响应表示 AT 已准备好接收 MQTT 用户名，此时你可以输入 MQTT 用户名，当 AT 接收到的 MQTT 用户名长度达到 <length> 后，返回：

OK

参数

- **<LinkID>**：当前仅支持 link ID 0。
- **<length>**：MQTT 用户名长度。范围：[1,1024]。

说明

- **AT+MQTTUSERCFG** 命令也可以设置 MQTT 用户名，二者之间的差别包括：
 - **AT+MQTTLONGUSERNAME** 命令可以用来设置相对较长的用户名，因为 **AT+MQTTUSERCFG** 命令的长度受限。
 - 应在设置 **AT+MQTTUSERCFG** 后再使用 **AT+MQTTLONGUSERNAME**。

3.5.5 AT+MQTTLONGPASSWORD：设置 MQTT 登陆密码

设置命令

功能：

设置 MQTT 密码

命令：

AT+MQTTLONGPASSWORD=<LinkID>,<length>

响应：

OK

>

上述响应表示 AT 已准备好接收 MQTT 密码，此时你可以输入 MQTT 密码，当 AT 接收到的 MQTT 密码长度达到 <length> 后，返回：

OK

参数

- **<LinkID>**：当前仅支持 link ID 0。
- **<length>**：MQTT 密码长度。范围：[1,1024]。

说明

- **AT+MQTTUSERCFG** 命令也可以设置 MQTT 密码，二者之间的差别包括：
 - **AT+MQTTLONGPASSWORD** 可以用来设置相对较长的密码，因为 **AT+MQTTUSERCFG** 命令的长度受限；
 - 应在设置 **AT+MQTTUSERCFG** 后再使用 **AT+MQTTLONGPASSWORD**。

3.5.6 AT+MQTTCONNCFG: 设置 MQTT 连接属性

设置命令

功能:

设置 MQTT 连接属性

命令:

```
AT+MQTTCONNCFG=<LinkID>,<keepalive>,<disable_clean_session>,<"lwt_topic">,<"lwt_msg"
↪">,<lwt_qos>,<lwt_retain>
```

响应:

OK

参数

- **<LinkID>**: 当前仅支持 link ID 0。
- **<keepalive>**: MQTT ping 超时时间, 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, 有关该参数的更多信息请参考 MQTT 3.1.1 协议中的 [Clean Session](#) 章节。
 - 0: 使能清理会话
 - 1: 禁用清理会话
- **<" lwt_topic" >**: 遗嘱 topic, 最大长度: 128 字节。
- **<" lwt_msg" >**: 遗嘱 message, 最大长度: 128 字节。
- **<lwt_qos>**: 遗嘱 QoS, 参数可选 0、1、2, 默认值: 0。
- **<lwt_retain>**: 遗嘱 retain, 参数可选 0 或 1, 默认值: 0。

说明

- 设置此命令前, 应先设置 [AT+MQTTUSERCFG](#)。

3.5.7 AT+MQTTALPN: 设置 MQTT 应用层协议协商 (ALPN)

设置命令

功能:

设置 MQTT 应用层协议协商 (ALPN)

命令:

```
AT+MQTTALPN=<LinkID>,<alpn_counts>[,<"alpn">][,<"alpn">][,<"alpn">]
```

响应:

OK

参数

- **<LinkID>**: 当前仅支持 link ID 0。
- **<alpn_counts>**: **<" alpn" >** 参数个数。范围: [0,5]。
 - 0: 清除 MQTT ALPN 配置
 - [1,5]: 设置 MQTT ALPN 配置
- **<" alpn" >**: 字符串参数, 表示 ClientHello 中的 ALPN, 用户可以发送多个 ALPN 字段到服务器。

说明

- 整条 AT 命令长度应小于 256 字节。
- 只有在 MQTT 基于 TLS 或 WSS 时，MQTT ALPN 字段才会生效。
- 应在设置 [AT+MQTTUSERCFG](#) 后再使用 [AT+MQTTALPN](#)。

示例

```
AT+CWMODE=1
AT+CWJAP="ssid", "password"
AT+CIPSNTPCFG=1,8,"ntp1.aliyun.com","ntp2.aliyun.com"
AT+MQTTUSERCFG=0,5,"ESP32-C6","espressif","1234567890",0,0,""
AT+MQTTALPN=0,2,"mqtt-ca.cn","mqtt-ca.us"
AT+MQTTCONN=0,"192.168.200.2",8883,1
```

3.5.8 AT+MQTTSNI: 设置 MQTT 服务器名称指示 (SNI)

设置命令

功能:

设置 MQTT 服务器名称指示 (SNI)

命令:

```
AT+MQTTSNI=<LinkID>,<"sni">
```

响应:

```
OK
```

参数

- **<LinkID>**: 当前仅支持 link ID 0。
- **<"sni">**: MQTT 服务器名称指示。你可以在 ClientHello 中将其发送到服务器。

说明

- 整条 AT 命令长度应小于 256 字节。
- 只有在 MQTT 基于 TLS 或 WSS 时，MQTT SNI 字段才会生效。
- 应在设置 [AT+MQTTUSERCFG](#) 后再使用 [AT+MQTTSNI](#)。

示例

```
AT+CWMODE=1
AT+CWJAP="ssid", "password"
AT+CIPSNTPCFG=1,8,"ntp1.aliyun.com","ntp2.aliyun.com"
AT+MQTTUSERCFG=0,5,"ESP32-C6","espressif","1234567890",0,0,""
AT+MQTTSNI=0,"my_specific_prefix.iot.my_aws_region.amazonaws.com"
AT+MQTTCONN=0,"my_specific_prefix.iot.my_aws_region.amazonaws.com",8883,1
```

3.5.9 AT+MQTTCONN: 连接 MQTT Broker

查询命令

功能:

查询 ESP32-C6 设备已连接的 MQTT broker

命令:

```
AT+MQTTCONN?
```

响应:

```
+MQTTCONN:<LinkID>,<state>,<scheme>,<"host">,<port>,<"path">,<reconnect>
OK
```

设置命令

功能:

连接 MQTT Broker

命令:

```
AT+MQTTCONN=<LinkID>,<"host">,<port>,<reconnect>[,<timeout_ms>]
```

响应:

```
OK
```

参数

- **<LinkID>**: 当前仅支持 link ID 0。
- **<" host" >**: MQTT broker 域名, 最大长度: 128 字节。
- **<port>**: MQTT broker 端口, 最大端口: 65535。
- **<" path" >**: 资源路径, 最大长度: 32 字节。
- **<reconnect>**:
 - 0: MQTT 不自动重连。如果 MQTT 建立连接后又断开, 则无法再次使用本命令重新建立连接, 你需要先发送 [AT+MQTTCLEAN=0](#) 命令清理信息, 重新配置参数, 再建立新的连接。
 - 1: MQTT 自动重连, 会消耗较多的内存资源。
- **<timeout_ms>**: 超时时间, 单位: 毫秒。范围: [3000,60000]。默认值: 15000 毫秒。
- **<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。
- **<scheme>**:
 - 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);
 - 6: MQTT over WebSocket (基于 TCP);
 - 7: MQTT over WebSocket Secure (基于 TLS, 不校验证书);
 - 8: MQTT over WebSocket Secure (基于 TLS, 校验 server 证书);

- 9: MQTT over WebSocket Secure (基于 TLS, 提供 client 证书);
- 10: MQTT over WebSocket Secure (基于 TLS, 校验 server 证书并且提供 client 证书)。

3.5.10 AT+MQTTPUB: 发布 MQTT 消息 (字符串)

设置命令

功能:

通过 topic 发布 MQTT 字符串消息。如果你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 字节, 请使用 [AT+MQTTPUBRAW](#) 命令。

命令:

```
AT+MQTTPUB=<LinkID>,<"topic">,<"data">,<qos>,<retain>[,<timeout_ms>]
```

响应:

```
OK
```

参数

- <LinkID>: 当前仅支持 link ID 0。
- <"topic">: MQTT topic, 最大长度: 128 字节。
- <"data">: MQTT 字符串消息。
- <qos>: 发布消息的 QoS, 参数可选 0、1、或 2, 默认值: 0。
- <retain>: 发布 retain。
- <timeout_ms>: 超时时间, 单位: 毫秒。范围: [3000,60000]。默认值: 15000 毫秒。

说明

- 每条 AT 命令的总长度不能超过 256 字节。
- 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。

示例

```
AT+CWMODE=1
AT+CWJAP="ssid","password"
AT+MQTTUSERCFG=0,1,"ESP32-C6","espressif","1234567890",0,0,""
AT+MQTTCONN=0,"192.168.10.234",1883,0
AT+MQTTPUB=0,"topic","\"{\\"timestamp\\":\\"20201121085253\\"}\"",0,0 //_
↪发送此命令时, 请注意特殊字符是否需要转义。
```

3.5.11 AT+MQTTPUBRAW: 发布长 MQTT 消息

设置命令

功能:

通过 topic 发布长 MQTT 消息。如果你发布消息的数据量相对较少, 不大于单条 AT 命令的长度阈值 256 字节, 也可以使用 [AT+MQTTPUB](#) 命令。

命令:

```
AT+MQTTPUBRAW=<LinkID>,<"topic">,<length>,<qos>,<retain>[,<timeout_ms>]
```

响应:

```
OK
>
```

符号 > 表示 AT 准备好接收串口数据，此时你可以输入数据，当数据长度达到参数 <length> 的值时，数据传输开始。

若传输成功，则 AT 返回：

```
+MQTTPUB:OK
```

若传输失败，则 AT 返回：

```
+MQTTPUB:FAIL
```

参数

- <LinkID>：当前仅支持 link ID 0。
- <" topic" >：MQTT topic，最大长度：128 字节。
- <length>：MQTT 消息长度，不同 ESP32-C6 设备的最大长度受到可利用内存的限制。
- <qos>：发布消息的 QoS，参数可选 0、1、或 2，默认值：0。
- <retain>：发布 retain。
- <timeout_ms>：超时时间，单位：毫秒。范围：[3000,60000]。默认值：15000 毫秒。

3.5.12 AT+MQTTSUB: 订阅 MQTT Topic**查询命令****功能:**

查询已订阅的 topic

命令:

```
AT+MQTTSUB?
```

响应:

```
+MQTTSUB:<LinkID>,<state>,<"topic1">,<qos>
+MQTTSUB:<LinkID>,<state>,<"topic2">,<qos>
+MQTTSUB:<LinkID>,<state>,<"topic3">,<qos>
...
OK
```

设置命令**功能:**

订阅指定 MQTT topic 的指定 QoS，支持订阅多个 topic（最多支持订阅 10 个 topic）。

命令:

```
AT+MQTTSUB=<LinkID>,<"topic">,<qos>[,<timeout_ms>]
```

响应:

```
OK
```

当 AT 接收到已订阅的 topic 的 MQTT 消息时，返回：

```
+MQTTSUBRECV:<LinkID>,<"topic">,<data_length>,<data>
```

若已订阅过该 topic，则返回：

```
ALREADY SUBSCRIBE
```

参数

- **<LinkID>**：当前仅支持 link ID 0。
- **<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 MQTT topic。
- **<"topic">**：订阅的 topic。
- **<qos>**：订阅的 QoS。
- **<timeout_ms>**：超时时间，单位：毫秒。范围：[3000,60000]。默认值：15000 毫秒。

说明

- 由于广域网每个路由节点的 MTU 限制或流量策略管理等，MQTT broker 发送的一条消息，最终可能会被分为多个消息发给 ESP32-C6 设备，因此 ESP32-C6 可能会收到多条 +MQTTSUBRECV 消息。
- 若 ESP32-C6 设备收到一条 MQTT 消息，且消息长度超过 1024 字节（包括 MQTT 报头和 MQTT 有效载荷），也会被分为多条 +MQTTSUBRECV 消息。此时，你可以通过下面方式增加 MQTT 缓冲区来避免此情况。
 - `./build.py menuconfig > Component config > ESP-MQTT Configurations > MQTT Using custom configurations`
 - `./build.py menuconfig > Component config > ESP-MQTT Configurations > MQTT Using custom configurations>Default MQTT Buffer Size>1460`

3.5.13 AT+MQTTUNSUB：取消订阅 MQTT Topic

设置命令

功能：

客户端取消订阅指定 topic，可多次调用本命令，以取消订阅不同的 topic。

命令：

```
AT+MQTTUNSUB=<LinkID>,<"topic">[,<timeout_ms>]
```

响应：

```
OK
```

若未订阅过该 topic，则返回：

```
NO UNSUBSCRIBE
```

```
OK
```

参数

- **<LinkID>**: 当前仅支持 link ID 0。
- **<" topic" >**: MQTT topic, 最大长度: 128 字节。
- **<timeout_ms>**: 超时时间, 单位: 毫秒。范围: [3000,60000]。默认值: 15000 毫秒。

3.5.14 AT+MQTTCLEAN: 断开 MQTT 连接

设置命令

功能:

断开 MQTT 连接, 释放资源。

命令:

```
AT+MQTTCLEAN=<LinkID>
```

响应:

```
OK
```

参数

- **<LinkID>**: 当前仅支持 link ID 0。

3.5.15 MQTT AT 错误码

MQTT 错误码以 `ERR CODE:0x<%08x>` 形式打印。

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

续下页

表 2 - 接上页

错误类型	错误码
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_KEEPA_LIVE_READ_FAILED	0x602B
AT_MQTT_KEEPA_LIVE_IS_NULL	0x602C
AT_MQTT_KEEPA_LIVE_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

续下页

表 2 - 接上页

错误类型	错误码
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

3.5.16 MQTT AT 说明

- 当 MQTT 连接断开时，会提示 +MQTTDISCONNECTED:<LinkID> 消息。
- 当 MQTT 连接建立时，会提示 +MQTTCONNECTED:<LinkID>,<scheme>,<"host">,port,<"path">,<reconnect> 消息。

3.6 HTTP AT 命令集

- 介绍
- [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](#): 设置/获取长的 HTTP URL
- [AT+HTTPCHEAD](#): 设置/查询 HTTP 请求头
- [AT+HTTPCFG](#): 设置 HTTP 客户端配置
- [HTTP AT 错误码](#)

3.6.1 介绍

重要：默认的 AT 固件支持此页面下的所有 AT 命令。如果你不需要 ESP32-C6 支持 HTTP 命令，请自行编译 [ESP-AT 工程](#)，在第五步配置工程里选择：

- 禁用 Component config>AT>AT http command support

3.6.2 AT+HTTPCLIENT: 发送 HTTP 客户端请求

设置命令

命令:

```
AT+HTTPCLIENT=<opt>,<content-type>,<"url">,[<"host">],[<"path">],<transport_type>[,
↩<"data">],[<"http_req_header">],[<"http_req_header">][...]
```

响应:

```
+HTTPCLIENT:<size>,<data>
```

```
OK
```

参数

- **<opt>**: HTTP 客户端请求方法:
 - 1: HEAD
 - 2: GET
 - 3: POST
 - 4: PUT
 - 5: DELETE
- **<content-type>**: 客户端请求数据类型:
 - 0: application/x-www-form-urlencoded
 - 1: application/json
 - 2: multipart/form-data
 - 3: text/xml
- **<" url">**: HTTP URL, 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。
- **<" path">**: HTTP 路径。
- **<transport_type>**: HTTP 客户端传输类型, 默认值为 1:
 - 1: HTTP_TRANSPORT_OVER_TCP
 - 2: HTTP_TRANSPORT_OVER_SSL
- **<" data">**: 当 **<opt>** 是 POST 请求时, 本参数为发送给 HTTP 服务器的数据。当 **<opt>** 不是 POST 请求时, 这个参数不存在 (也就是, 不需要输入逗号来表示有这个参数)。
- **<" http_req_header">**: 可发送多个请求头给服务器。

说明

- 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。
- 某些已发布的固件默认不支持 HTTP 客户端命令 (详情请见 [ESP-AT 固件差异](#)), 但是可通过以下方式使其支持该命令: `./build.py menuconfig > Component config > AT > AT http command support`, 然后编译项目 (详情请见 [本地编译 ESP-AT 工程](#))。
- 该命令不支持 URL 重定向, 在获取到服务器返回的状态码 301 (永久性重定向) 或者 302 (临时性重定向) 后不会自动跳转到新的 URL 地址。你可以使用某些工具获取要访问的实际 URL, 然后通过该命令访问它。
- 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。

示例

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

// GET 请求
AT+HTTPCLIENT=2,0,"http://httpbin.org/get","httpbin.org","/get",1

// POST 请求
AT+HTTPCLIENT=3,0,"http://httpbin.org/post","httpbin.org","/post",1,"field1=value1&
↪field2=value2"
```

3.6.3 AT+HTTPGETSIZE: 获取 HTTP 资源大小

设置命令

命令:

```
AT+HTTPGETSIZE=<"url">[,<tx size>][,<rx size>][,<timeout>]
```

响应:

```
+HTTPGETSIZE:<size>
```

```
OK
```

参数

- <"url">: HTTP URL。
- <tx size>: HTTP 发送缓存大小。单位: 字节。默认值: 2048。范围: [0,10240]。
- <rx size>: HTTP 接收缓存大小。单位: 字节。默认值: 2048。范围: [0,10240]。
- <timeout>: 网络超时。单位: 毫秒。默认值: 5000。范围: [0,180000]。
- <size>: HTTP 资源大小。

说明

- 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 ""。
- 如果你想设置 HTTP 请求头, 请使用 [AT+HTTPHEAD](#) 命令设置。

示例

```
AT+HTTPGETSIZE="http://www.baidu.com/img/bdlogo.gif"
```

3.6.4 AT+HTTPGET: 获取 HTTP 资源

设置命令

命令:

```
AT+HTTPGET=<"url">[,<tx size>][,<rx size>][,<timeout>]
```

响应:

```
+HTTPGET:<size>,<data>
```

```
OK
```

参数

- <"url">: HTTP URL。
- <tx size>: HTTP 发送缓存大小。单位: 字节。默认值: 2048。范围: [0,10240]。
- <rx size>: HTTP 接收缓存大小。单位: 字节。默认值: 2048。范围: [0,10240]。
- <timeout>: 网络超时。单位: 毫秒。默认值: 5000。范围: [0,180000]。

说明

- 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 ""。
- 如果你想设置 HTTP 请求头, 请使用 [AT+HTTPHEAD](#) 命令设置。

3.6.5 AT+HTTPCPOST: Post 指定长度的 HTTP 数据

设置命令

命令:

```
AT+HTTPCPOST=<"url">,<length>[,<http_req_header_cnt>][,<"http_req_header">..<"http_req_header">]
```

响应:

```
OK
```

```
>
```

符号 > 表示 AT 准备好接收串口数据，此时你可以输入数据，当数据长度达到参数 <length> 的值时，传输开始。

若传输成功，则返回：

```
SEND OK
```

若传输失败，则返回：

```
SEND FAIL
```

参数

- <"url">: HTTP URL。
- <length>: 需 POST 的 HTTP 数据长度。最大长度等于系统可分配的堆空间大小。
- <http_req_header_cnt>: <"http_req_header"> 参数的数量。
- <"http_req_header">: HTTP 请求头。可发送多个请求头给服务器。

说明

- 如果包含 URL 的整条命令的长度超过了 256 字节，请先使用 [AT+HTTURLCFG](#) 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 content-type 默认类型为 application/x-www-form-urlencoded。
- 如果你想设置 HTTP 请求头，请使用 [AT+HTTPCHEAD](#) 命令设置。
- 性能优化：本命令使用短连接模式，每次请求都会建立新连接（HTTPS 需进行 SSL 握手）。为实现低时延通信，建议改用 [AT+CIPSTART](#) 建立长连接，由 MCU 构造 HTTP 请求包并通过 [AT+CIPSEND](#) 发送，以避免重复握手产生的通信开销。
- 内存管理：本命令每次新建并关闭连接，会产生 TIME_WAIT 状态占用内存。若发现最小堆空间持续下降，可通过以下方式优化：
 - 减小 TCP Maximum Segment Lifetime (MSL): ./build.py menuconfig > Component config > LWIP > TCP > Maximum segment lifetime (MSL)，然后重新编译 AT 固件。注意：减小 MSL 可以缩短 TIME_WAIT 状态的持续时间，从而更快释放内存，但不能完全消除 TIME_WAIT 状态。
 - 改用 TCP 长连接复用，避免频繁创建和关闭连接。

3.6.6 AT+HTTPCPUT: Put 指定长度的 HTTP 数据

设置命令

命令:

```
AT+HTTPCPUT=<"url">,<length>[,<http_req_header_cnt>][,<"http_req_header">..<"http_req_header">]
```

响应:

```
OK
>
```

符号 > 表示 AT 准备好接收串口数据，此时你可以输入数据，当数据长度达到参数 <length> 的值时，传输开始。

若传输成功，则返回：

```
SEND OK
```

若传输失败，则返回：

```
SEND FAIL
```

参数

- <"url">: HTTP URL。
- <length>: 需 Put 的 HTTP 数据长度。最大长度等于系统可分配的堆空间大小。
- <http_req_header_cnt>: <"http_req_header"> 参数的数量。
- <"http_req_header">: HTTP 请求头。可发送多个请求头给服务器。

说明

- 如果包含 URL 的整条命令的长度超过了 256 字节，请先使用 [AT+HTTURLCFG](#) 命令预配置 URL，然后本命令里的 <"url"> 参数需要设置为 ""。
- 如果你想设置 HTTP 请求头，请使用 [AT+HTTPHEAD](#) 命令设置。

3.6.7 AT+HTTURLCFG: 设置/获取长的 HTTP URL**查询命令****命令:**

```
AT+HTTURLCFG?
```

响应:

```
[+HTTURLCFG:<url length>,<data>]
OK
```

设置命令**命令:**

```
AT+HTTURLCFG=<url length>
```

响应:

OK

>

符号 > 表示 AT 准备好接收串口数据，此时你可以输入 URL，当数据长度达到参数 <url length> 的值时，系统返回：

SET OK

参数

- <url length>: HTTP URL 长度。单位：字节。
 - 0: 清除 HTTP URL 配置。
 - [8,8192]: 设置 HTTP URL 配置。
- <data>: HTTP URL 数据。

3.6.8 AT+HTTPCHEAD: 设置/查询 HTTP 请求头

查询命令

命令:

AT+HTTPCHEAD?

响应:

+HTTPCHEAD:<index>,<"req_header">

OK

设置命令

命令:

AT+HTTPCHEAD=<req_header_len>

响应:

OK

>

符号 > 表示 AT 准备好接收 AT 命令口数据，此时你可以输入 HTTP 请求头（请求头为 key: value 形式，无需以 \r\n 结尾），当数据长度达到参数 <req_header_len> 的值时，AT 返回：

OK

参数

- <index>: HTTP 请求头的索引值。
- <" req_header" >: HTTP 请求头。
- <req_header_len>: HTTP 请求头长度。单位：字节。
 - 0: 清除所有已设置的 HTTP 请求头。
 - 其他值: 设置一个新的 HTTP 请求头。

说明

- 本命令一次只能设置一个 HTTP 请求头，但可以多次设置，支持多个不同的 HTTP 请求头。
- 本命令配置的 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。
- 本命令多次设置的 HTTP 请求头中的 key 如果有重复，则会使用最后一次设置的值。

示例

```
// 设置请求头
AT+HTTPCHEAD=18

// 在收到 ">" 符号后，输入以下的 Range 请求头，下载资源的前 256 个字节。
Range: bytes=0-255

// 下载 HTTP 资源
AT+HTTPCGET="https://docs.espressif.com/projects/esp-at/zh_CN/latest/esp32c6/index.
↵html"
```

3.6.9 AT+HTTPCFG：设置 HTTP 客户端配置

设置命令

命令：

```
AT+HTTPCFG=<auth_mode>[,<pki_number>][,<ca_number>]
```

响应：

```
OK
```

参数

- **<auth_mode>**:
 - 0: 不认证，此时无需填写 <pki_number> 和 <ca_number> 参数；
 - 1: ESP-AT 提供 HTTP 客户端证书供 HTTP 服务器端 CA 证书校验；
 - 2: ESP-AT HTTP 客户端载入 CA 证书来校验 HTTP 服务器端的证书；
 - 3: 相互认证。
- **<pki_number>**: 证书和私钥的索引，如果只有一个证书和私钥，其值应为 0。
- **<ca_number>**: CA 的索引，如果只有一个 CA，其值应为 0。

说明

- 默认 AT 固件不支持 HTTP 证书配置，你可以启用 `./build.py menuconfig > Component config > AT > AT http command support > AT HTTP authentication method` 下选型来使其支持。
- 本命令配置的参数是全局性的，一旦设置，所有 HTTP 命令都会共用该配置。
- 如果你想使用自己的证书，运行时请使用 `AT+SYSMFG` 命令更新 HTTP 证书（具体步骤请参考 `AT+SYSMFG` 命令示例，证书配置方法与 SSL 证书相同）。如果你想预烧录自己的证书，请参考 `如何更新 PKI 配置`。
- 如果 <auth_mode> 配置为 2 或者 3，为了校验服务器的证书有效期，请在发送其它 HTTP 命令前确保 ESP32-C6 已获取到当前时间（你可以发送 `AT+CIPSNTPCFG` 命令来配置 SNTP，获取当前时间，发送 `AT+CIPSNTPTIME?` 命令查询当前时间）。

3.6.10 HTTP AT 错误码

启用 `AT+SYSLOG=1` 后，HTTP 客户端请求失败时，AT 会返回错误码。错误码的格式为：

```
ERR CODE:0x010a7xxx
```

其中 01 是模块标识符，0a 是模块执行 AT 命令的响应结果，7xxx 表示 HTTP 错误码。如果 xxx 在 HTTP 标准状态码范围内，则表示标准 HTTP 状态码；否则表示 AT HTTP 内部的错误码。下表列出了部分 HTTP 状态码信息，更多详情请参考 [RFC 2616](#)。

HTTP 错误码 (HTTP 状态码)	说明
0x7000	建立连接失败
0x7190 (400)	Bad Request
0x7191 (401)	Unauthorized
0x7192 (402)	Payment Required
0x7193 (403)	Forbidden
0x7194 (404)	Not Found
0x7195 (405)	Method Not Allowed
0x7196 (406)	Not Acceptable
0x7197 (407)	Proxy Authentication Required
0x7198 (408)	Request Timeout
0x7199 (409)	Conflict
0x719a (410)	Gone
0x719b (411)	Length Required
0x719c (412)	Precondition Failed
0x719d (413)	Request Entity Too Large
0x719e (414)	Request-URI Too Long
0x719f (415)	Unsupported Media Type
0x71a0 (416)	Requested Range Not Satisfiable
0x71a1 (417)	Expectation Failed
0x71f4 (500)	Internal Server Error
0x71f5 (501)	Not Implemented
0x71f6 (502)	Bad Gateway
0x71f7 (503)	Service Unavailable
0x71f8 (504)	Gateway Timeout
0x71f9 (505)	HTTP Version Not Supported

3.7 文件系统 AT 命令集

- [介绍](#)
- `AT+FS`：文件系统操作
- `AT+FSMOUNT`：挂载/卸载文件系统

3.7.1 介绍

重要：默认的 AT 固件不支持此页面下的 AT 命令。如果你需要 ESP32-C6 支持文件系统命令，请自行编译 `ESP-AT` 工程，在第五步配置工程里选择：

- 启用 Component config>AT>AT FS command support

3.7.2 AT+FS：文件系统操作

设置命令

命令：

```
AT+FS=<type>,<operation>,<filename>,<offset>,<length>
```

响应：

```
OK
```

参数

- **<type>**：文件系统类型，推荐使用 LittleFS。可在 Component config > AT > AT FS command support > Filesystem type for AT FS command 中选择。
 - 0: FATFS
 - 1: LittleFS
- **<operation>**：
 - 0: 删除文件
 - 1: 写文件
 - 2: 读文件
 - 3: 查询文件大小
 - 4: 查询路径下文件，目前仅支持根目录
 - 5: 计算文件的 MD5 哈希值，用于校验文件完整性
 - 6: 查询文件系统的空闲空间和总空间
- **<filename>**：文件或目录名称，仅操作 0 至 5 时需要此参数
- **<offset>**：从文件起始位置的字节偏移量，仅针对读写操作设置
- **<length>**：数据长度，仅针对读写操作设置

说明

- 本命令会自动挂载文件系统。**AT+FS** 文件系统操作完成后，强烈建议使用 **AT+FSMOUNT=0** 命令卸载文件系统，来释放大容量 RAM 空间。
- 使用本命令需烧录 at_customize.bin，详细信息可参考 [ESP-IDF 分区表](#) 和 [如何自定义分区](#)。
- 若读取数据的长度大于实际文件大小，仅返回实际长度的数据。
- 当 <operation> 为 write 时，系统收到此命令后先换行返回 >，此时你可以输入要写的的数据，数据长度应与 <length> 一致。
- 若需通过 OTA 升级 AT 固件并确保其兼容旧版 FATFS 分区，请在以下配置路径 Component config > AT > AT FS command support > Filesystem type for AT FS command 中选择 FatFS (legacy partition, upgrade only, READ HELP FIRST)。

示例

```
// 删除某个文件
AT+FS=1,0,"filename"

// 在某个文件偏移地址 100 处写入 10 字节
AT+FS=1,1,"filename",100,10

// 从某个文件偏移地址 0 处读取 100 字节
AT+FS=1,2,"filename",0,100

// 列出根目录下所有文件
AT+FS=1,4,"."
```

(续下页)

(接上页)

```
// 计算某个文件的 MD5 哈希值
AT+FS=1,5,"filename"

// 查询文件系统的空闲空间和总空间
AT+FS=1,6
```

3.7.3 AT+FSMOUNT: 挂载/卸载 FS 文件系统

设置命令

命令:

```
AT+FSMOUNT=<mount>
```

响应:

```
OK
```

参数

- **<mount>**:
 - 0: 卸载 FS 文件系统
 - 1: 挂载 FS 文件系统

说明

- **AT+FS** 文件系统操作完成后, 强烈建议使用 **AT+FSMOUNT=0** 命令卸载文件系统, 来释放大量的 RAM 空间。

示例

```
// 手动卸载文件系统
AT+FSMOUNT=0

// 手动挂载文件系统
AT+FSMOUNT=1
```

3.8 WebSocket AT 命令集

- [介绍](#)
- [AT+WSCFG](#): 配置 WebSocket 参数
- [AT+WSHEAD](#): 设置/查询 WebSocket 请求头
- [AT+WSOOPEN](#): 查询/打开 WebSocket 连接
- [AT+WSEND](#): 向 WebSocket 连接发送数据
- [AT+WSDATAFMT](#): 设置 WebSocket 接收数据格式
- [AT+WSCLOSE](#): 关闭 WebSocket 连接
- [WebSocket 示例](#)

3.8.1 介绍

重要： 默认的 AT 固件不支持此页面下的 AT 命令。如果你需要 ESP32-C6 支持 WebSocket 命令，请自行编译 [ESP-AT 工程](#)，在第五步配置工程里选择：

- 启用 Component config>AT>AT WebSocket command support

3.8.2 AT+WSCFG：配置 WebSocket 参数

设置命令

命令：

```
AT+WSCFG=<link_id>,<ping_intv_sec>,<ping_timeout_sec>[,<buffer_size>][,<auth_mode>,<pkc_number>,<ca_number>]
```

响应：

OK

或

ERROR

参数

- **<link_id>**：WebSocket 连接 ID。范围：[0,2]，即最大支持三个 WebSocket 连接。
- **<ping_intv_sec>**：发送 WebSocket Ping 间隔。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。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。单位：字节。范围：[1,16384]。默认值：1024。
- **<auth_mode>**：
 - 0: 不认证，此时无需填写 <pkc_number> 和 <ca_number> 参数；
 - 1: ESP-AT 提供客户端证书供服务器端 CA 证书校验；
 - 2: ESP-AT 客户端载入 CA 证书来校验服务器端的证书；
 - 3: 相互认证。
- **<pkc_number>**：证书和私钥的索引，如果只有一个证书和私钥，其值应为 0。
- **<ca_number>**：CA 的索引，如果只有一个 CA，其值应为 0。

说明

- 此命令应在 [AT+WSOPEN](#) 之前配置，否则不会生效。
- 如果你想使用自己的证书，运行时请使用 [AT+SYSMF](#) 命令更新 WebSocket 证书。如果你想预烧录自己的证书，请参考 [如何更新 PKI 配置](#)。
- 如果 <auth_mode> 配置为 2 或者 3，为了校验服务器的证书有效期，请在发送 [AT+WSOPEN](#) 命令前确保 ESP32-C6 已获取到当前时间（你可以发送 [AT+CIPSNTPCFG](#) 命令来配置 SNTP，获取当前时间，发送 [AT+CIPSNTPTIME?](#) 命令查询当前时间）。
- 相互认证的示例：基于 [TLS](#) 的 [WebSocket](#) 连接（相互鉴权）。

示例


```
// 配置 link_id 为 0 的 WebSocket 连接的 Ping 发送间隔为 30 秒，超时 60 秒，缓冲区
↪4096 字节
AT+WSCFG=0,30,60,4096
```

3.8.3 AT+WSHEAD: 设置/查询 WebSocket 请求头

查询命令

命令:

```
AT+WSHEAD?
```

响应:

```
+WSHEAD:<index>,<"req_header">
```

```
OK
```

设置命令

命令:

```
AT+WSHEAD=<req_header_len>
```

响应:

```
OK
```

```
>
```

符号 > 表示 AT 准备好接收 AT 命令口数据，此时你可以输入 WebSocket 请求头（请求头为 key: value 形式，无需以 \r\n 结尾），当数据长度达到参数 <req_header_len> 的值时，AT 返回：

```
OK
```

参数

- **<index>**: WebSocket 请求头的索引值。
- **<" req_header" >**: WebSocket 请求头。
- **<req_header_len>**: WebSocket 请求头长度。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。

说明

- 本命令一次只能设置一个 WebSocket 请求头，但可以多次设置，支持多个不同的 WebSocket 请求头。
- 本命令配置的 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。

示例

```
// 设置请求头
AT+WSHEAD=49

// 在收到 ">" 符号后，输入以下的 authorization 请求头
AUTHORIZATION: Basic QTIzMzIyMDE5OTk6MTIzNDU2Nzg=

// 打开一个 WebSocket 连接
AT+WSOPEN=0,"wss://demo.piesocket.com/v3/channel_123?api_
↪key=VCXCEuvhGcBDP7XhiJJUDvR1e1D3eiVjgZ9VRiaV&notify_self"
```

3.8.4 AT+WSOPEN: 查询/打开一个 WebSocket 连接

查询命令

命令:

```
AT+WSOPEN?
```

响应:

当有连接时，AT 返回:

```
+WSOPEN:<link_id>,<state>,<"uri">
```

```
OK
```

当没有连接时，AT 返回:

```
OK
```

设置命令

命令:

```
AT+WSOPEN=<link_id>,<"uri">[,<"subprotocol">][,<timeout_ms>][,<"auth">]
```

响应:

```
+WS_CONNECTED:<link_id>
```

```
OK
```

或

```
ERROR
```

参数

- **<link_id>**: WebSocket 连接 ID。范围: [0,2]，即最大支持三个 WebSocket 连接。
- **<state>**: WebSocket 连接的状态。
 - 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。
 - 4: 已收到服务器端 WebSocket 关闭帧，正在发送关闭帧到服务器。
- **<" uri" >**: WebSocket 服务器的统一资源标识符。
- **<" subprotocol" >**: WebSocket 子协议 (参考 [RFC6455 1.9 章节](#))。

- **<timeout_ms>**: 建立 WebSocket 连接的超时时间。单位: 毫秒。范围: [0,180000]。默认值: 15000。
- **<"auth">**: WebSocket 鉴权 (参考 [RFC6455 4.1.12 章节](#))。

示例

```
// uri 参数来自于 https://www.piesocket.com/websocket-tester
AT+WSOOPEN=0,"wss://demo.piesocket.com/v3/channel_123?api_
↪key=VCXCEuvhGcBDP7XhiJJUDvR1e1D3eiVjgZ9VRiaV&notify_self"
```

详细示例参考: [WebSocket 示例](#)。

3.8.5 AT+WSEND: 向 WebSocket 连接发送数据

设置命令

命令:

```
AT+WSEND=<link_id>,<length>[,<opcode>][,<timeout_ms>][,<fin>]
```

响应:

```
OK
>
```

上述响应表示 AT 已准备好从 AT port 接收数据, 此时你可以输入数据, 当 AT 接收到的数据长度达到 <length> 后, 数据传输开始。

如果未建立连接或数据传输时连接被断开, 返回:

```
ERROR
```

如果数据传输成功, 返回:

```
SEND OK
```

参数

- **<link_id>**: WebSocket 连接 ID。范围: [0,2]。
- **<length>**: 发送的数据长度。单位: 字节。最大可发送长度取决于 16 KiB 与系统可分配堆空间大小中的较小者。
- **<opcode>**: 发送的 WebSocket 帧中的 opcode。范围: [0,0xF]。默认值: 1, 即 text 帧。请参考 [RFC6455 5.2 章节](#) 了解更多的 opcode。
 - 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。单位: 毫秒。范围: [0,60000]。默认值: 10000。
- **<fin>**: 表示数据帧是否为最后一个分片。默认值: 1。
 - 0: 表示数据帧还有后续分片。
 - 1: 表示数据帧为最后一个分片。

3.8.6 AT+WSDATAFMT: 设置 WebSocket 接收数据格式

设置命令

功能:

默认情况下, ESP32-C6 在收到 WebSocket 数据时, 会以 +WS_DATA:<link_id>,<data_len>,<data>\r\n 的格式输出数据。本命令允许你配置接收数据时的元数据输出格式, 以满足不同的应用需求。完整的输出格式如下:

```
+WS_DATA:<link_id>[,<opcode>][,<fin>][,<payload_len>][,<payload_offset>][,<data_
len>],<data>[<\r\n>]
```

命令:

```
AT+WSDATAFMT=<link_id>[,<meta_data_level>][,<meta_data_mask>]
```

响应:

```
OK
```

或

```
ERROR
```

参数

- **<link_id>**: WebSocket 连接 ID。范围: [0,2]。
- **<meta_data_level>**: 元数据输出等级, 默认值: 0。各等级含义如下:
 - 0: chunked 等级。每次从传输层收到数据, 都会添加 +WS_DATA 开头的元数据。
 - 1: frame 等级。仅在完整的 WebSocket 帧前后, 会添加 +WS_DATA 开头的元数据。
 - 2: message 等级。仅在完整的 WebSocket 消息前后, 会添加 +WS_DATA 开头的元数据。
- **<meta_data_mask>**: 用于控制 +WS_DATA 输出格式的位掩码。将相应 bit 设为 1 即可在输出中包含该位所对应的字段。默认值: 48 (此时仅输出 <data_len> 和 <\r\n>)。
 - bit 0: <opcode>, WebSocket 帧类型
 - bit 1: <fin>, 是否为消息的最后一帧
 - bit 2: <payload_len>, 完整负载长度
 - bit 3: <payload_offset>, 当前数据在负载中的偏移
 - bit 4: <data_len>, 当前输出的数据长度
 - bit 5: <\r\n>, 数据末尾的换行符

示例

```
// 设置 link_id 为 0 的 WebSocket 连接, 使用 message 等级, 仅输出 payload_len。
AT+WSDATAFMT=0,2,4
```

3.8.7 AT+WSCLOSE: 关闭 WebSocket 连接

设置命令

命令:

```
AT+WSCLOSE=<link_id>
```

响应:

OK

参数

- **<link_id>**: WebSocket 连接 ID。范围: [0,2]。

说明

- 本命令触发 TCP 层的关闭流程, 返回 OK 仅表示关闭指令已传达, 不保证对端已完全断开连接。若需确保连接完全关闭, 建议在执行本命令前, 先使用 [AT+WSEND](#) 发送一个 opcode 为 8 (Close Frame) 的 WebSocket 控制帧, 等待服务器回应后再关闭。

示例

// 关闭 ID 为 0 的 WebSocket 连接
AT+WSCLOSE=0

3.9 信令测试 AT 命令

- [介绍](#)
- [AT+FACTPLCP](#): 发送长 PLCP 或短 PLCP

3.9.1 介绍

重要: 默认的 AT 固件支持此页面下的所有 AT 命令。如果你不需要 ESP32-C6 支持信令测试命令, 请自行编译 [ESP-AT 工程](#), 在第五步配置工程里选择:

- 禁用 Component config>AT>AT signaling test command support
-

3.9.2 AT+FACTPLCP: 发送长 PLCP 或短 PLCP

设置命令

命令:

AT+FACTPLCP=<enable>,<tx_with_long>

响应:

OK

参数

- **<enable>**: 启用或禁用手动配置:
 - 0: 禁用手动配置, 将使用 <tx_with_long> 参数的默认值;
 - 1: 启用手动配置, AT 发送的 PLCP 类型取决于 <tx_with_long> 参数。
- **<tx_with_long>**: 发送长 PLCP 或短 PLCP:

- 0: 发送短 PLCP (默认);
- 1: 发送长 PLCP。

3.10 驱动 AT 命令

- [介绍](#)
- [AT+DRVADC](#): 读取 ADC 通道值
- [AT+DRVPWMINIT](#): 初始化 PWM 驱动器
- [AT+DRVPWMDUTY](#): 设置 PWM 占空比
- [AT+DRVPWMFADE](#): 设置 PWM 渐变
- [AT+DRVI2CINIT](#): 初始化 I2C 主机驱动
- [AT+DRVI2CRD](#): 读取 I2C 数据
- [AT+DRVI2CWRDATA](#): 写入 I2C 数据
- [AT+DRVI2CWRBYTES](#): 写入不超过 4 字节的 I2C 数据
- [AT+DRVSPICONFGPIO](#): 配置 SPI GPIO
- [AT+DRVSPIINIT](#): 初始化 SPI 主机驱动
- [AT+DRVSPIRD](#): 读取 SPI 数据
- [AT+DRVSPIWR](#): 写入 SPI 数据

3.10.1 介绍

重要: 默认的 AT 固件不支持此页面下的 AT 命令。如果你需要 ESP32-C6 支持驱动命令, 请自行[编译 ESP-AT 工程](#), 在第五步配置工程里选择:

- 启用 Component config>AT>AT driver command support

3.10.2 AT+DRVADC: 读取 ADC 通道值

设置命令

命令:

```
AT+DRVADC=<channel>,<atten>
```

响应:

```
+DRVADC:<raw data>
```

```
OK
```

参数

- **<channel>**: ADC1 通道。
 - ESP32-C6 设备的取值范围为 [0,4]。

通道	管脚
0	GPIO0
1	GPIO1
2	GPIO2
3	GPIO3
4	GPIO4

- **<atten>**: 衰减值。
 - 0: 0 dB 衰减, 有效测量范围为 [0, 750] mV。
 - 1: 2.5 dB 衰减, 有效测量范围为 [0, 1050] mV。
 - 2: 6 dB 衰减, 有效测量范围为 [0, 1300] mV。
 - 3: 11 dB 衰减, 有效测量范围为 [0, 2500] mV。
- **<raw data>**: ADC 通道值。

说明

- ESP-AT 只支持 ADC1。
- ESP32-C6 支持 12 位宽度。
- 对于如何将通道值转换为电压, 可以参考 [ADC 转换](#)。

示例

```
// ESP32-C6 设备设置为 0 dB 衰减, 有效测量范围为 [0, 750] mV
// 电压为 2048 / 4095 * 750 = 375.09 mV
AT+DRVADC=0,0
+DRVADC:2048

OK
```

3.10.3 AT+DRVPWMINIT: 初始化 PWM 驱动器

设置命令

命令:

```
AT+DRVPWMINIT=<freq>,<duty_res>,<ch0_gpio>[,...,<ch3_gpio>]
```

响应:

```
OK
```

参数

- **<freq>**: LEDC 定时器频率, 单位: Hz, 范围: 1 Hz ~ 8 MHz。
- **<duty_res>**: LEDC 通道占空比分辨率, 范围: 0 ~ 20 位。
- **<chx_gpio>**: LEDC 通道 x 的输出 GPIO。例如, 如果你想将 GPIO16 作为通道 0, 需设置 <ch0_gpio> 为 16。

说明

- ESP-AT 最多能支持 4 个通道。
- 使用本命令初始化的通道数量直接决定了其它 PWM 命令 (如 [AT+DRVPWMDUTY](#) 和 [AT+DRVPWMFADE](#)) 能够设置的通道。例如, 如果你只初始化了两个通道, 那么 AT+DRVPWMDUTY 命令只能用来更改这两个通道的 PWM 占空比。
- 频率和占空比分辨率相互影响。更多信息请见 [频率和占空比分辨率支持范围](#)。

示例

```
AT+DRVPWMINIT=5000,13,17,16,18,19 // 设置 4 个通道，频率为 5 kHz，占空比分辨率为 13 位
AT+DRVPWMINIT=10000,10,17 // 只初始化通道 0，频率为 10 kHz，占空比分辨率为 10 位，其它 PWM 相关命令只能设置一个通道
```

3.10.4 AT+DRVPWMDUTY：设置 PWM 占空比

设置命令

命令：

```
AT+DRVPWMDUTY=<ch0_duty>[,...,<ch3_duty>]
```

响应：

```
OK
```

参数

- <duty>：LEDC 通道占空比，范围：[0,2^{占空比分辨率}]。

说明

- ESP-AT 最多能支持 4 个通道。
- 若某个通道无需设置占空比，直接省略该参数。

示例

```
AT+DRVPWMDUTY=255,512 // 设置通道 0 的占空比为 255，设置通道 1 的占空比为 512
AT+DRVPWMDUTY=,,0 // 只设置通道 2 的占空比为 0
```

3.10.5 AT+DRVPWMFADE：设置 PWM 渐变

设置命令

命令：

```
AT+DRVPWMFADE=<ch0_target_duty>,<ch0_fade_time>[,...,<ch3_target_duty>,<ch3_fade_time>]
```

响应：

```
OK
```

参数

- <target_duty>：目标渐变占空比，范围：[0,2^{duty_resolution-1}]。
- <fade_time>：渐变的最长时间，单位：毫秒。

说明

- ESP-AT 最多能支持 4 个通道。
- 若某个通道无需设置 <target_duty> 和 <fade_time>, 直接省略即可。

示例

```
AT+DRVPWMFADE=,,0,1000 // 使用一秒的时间将通道 1 的占空比设置为 0
AT+DRVPWMFADE=1024,1000,0,2000, // 使用一秒的时间将通道 0 的占空比设置为 1024、两秒的时间将通道 1 的占空比设为 0
```

3.10.6 AT+DRVI2CINIT: 初始化 I2C 主机驱动

设置命令

命令:

```
AT+DRVI2CINIT=<num>,<scl_io>,<sda_io>,<clock>
```

响应:

```
OK
```

参数

- <num>: I2C 端口号, 范围: 0 ~ 1。如果未设置后面的参数, AT 将不初始化该 I2C 端口。
- <scl_io>: I2C SCL 信号的 GPIO 号。
- <sda_io>: I2C SDA 信号的 GPIO 号。
- <clock>: 主机模式下的 I2C 时钟频率, 单位: Hz, 最大值: 1 MHz。

说明

- 本命令只支持 I2C 主机。

示例

```
AT+DRVI2CINIT=0,25,26,1000 // 初始化 I2C0, SCL: GPIO25, SDA: GPIO26, I2C 时钟频率: 1 kHz
AT+DRVI2CINIT=0 // 取消 I2C0 初始化
```

3.10.7 AT+DRVI2CRD: 读取 I2C 数据

设置命令

命令:

```
AT+DRVI2CRD=<num>,<address>,<length>
```

响应:

```
+DRVI2CRD:<read data>
OK
```

参数

- **<num>**: I2C 端口号, 范围: 0 ~ 1。
- **<address>**: I2C 从机设备地址:
 - 7 位地址: 0 ~ 0x7F;
 - 10 位地址: 第一个字节的前七个位是 1111 0XX, 其中最后两位 XX 是 10 位地址的最高两位。例如, 如果 10 位地址为 0x2FF (b' 101111111), 那么输入的地址为 0x7AFF (b' 11110101111111)。
- **<length>**: I2C 数据长度, 范围: 1 ~ 2048。
- **<read data>**: I2C 数据。

说明

- I2C 传输超时时间为一秒。

示例

```
AT+DRVI2CRD=0,0x34,1      // I2C0 从地址 0x34 处读取 1 字节的数据
AT+DRVI2CRD=0,0x7AFF,1    // I2C0 从 10 位地址 0x2FF 处读取 1 字节的数据

// I2C0 读地址 0x34, 寄存器地址 0x27, 读 2 字节
AT+DRVI2CWRBYTES=0,0x34,1,0x27    // I2C0 先写设备地址 0x34、寄存器地址 0x27
AT+DRVI2CRD=0,0x34,2              // I2C0 读地址 2 字节
```

3.10.8 AT+DRVI2CWRDATA: 写入 I2C 数据

设置命令

命令:

```
AT+DRVI2CWRDATA=<num>,<address>,<length>
```

响应:

```
OK
>
```

收到上述响应后, 请输入你想写入的数据, 当数据达到参数指定长度后, 数据传输开始。

若数据传输成功, 则返回:

```
OK
```

若数据传输失败, 则返回:

```
ERROR
```

参数

- **<num>**: I2C 端口号, 范围: 0 ~ 1。
- **<address>**: I2C 从机设备地址:
 - 7 位地址: 0 ~ 0x7F;
 - 10 位地址: 第一个字节的前七个位是 1111 0XX, 其中最后两位 XX 是 10 位地址的最高两位。例如, 如果 10 位地址为 0x2FF (b' 101111111), 那么输入的地址为 0x7AFF (b' 11110101111111)。
- **<length>**: I2C 数据长度, 范围: 1 ~ 2048。

说明

- I2C 传输超时时间为一秒。

示例

```
AT+DRVI2CWRDATA=0,0x34,10 // I2C0 写入 10 字节数据至地址 0x34
```

3.10.9 AT+DRVI2CWRBYTES: 写入不超过 4 字节的 I2C 数据

设置命令

命令:

```
AT+DRVI2CWRBYTES=<num>,<address>,<length>,<data>
```

响应:

```
OK
```

参数

- **<num>**: I2C 端口号, 范围: 0~1。
- **<address>**: I2C 从机设备地址。
 - 7 位地址: 0~0x7F。
 - 10 位地址: 第一个字节的前七个位是 11110XX, 其中最后两位 XX 是 10 位地址的最高两位。例如, 如果 10 位地址为 0x2FF (b' 101111111), 那么输入的地址为 0x7AFF (b' 11110101111111)。
- **<length>**: 待写入的 I2C 数据长度, 范围: 1~4 字节。
- **<data>**: 参数 <length> 指定长度的数据, 范围: 0~0xFFFFFFFF。

说明

- I2C 传输超时时间为一秒。

示例

```
AT+DRVI2CWRBYTES=0,0x34,2,0x1234 // I2C0 写入 2 字节数据 0x1234 至地址 0x34
AT+DRVI2CWRBYTES=0,0x7AFF,2,0x1234 // I2C0 写入 2 字节数据 0x1234 至 10 位地址 0x7AFF
↪ 0x2FF

// I2C0 写地址 0x34、寄存器地址 0x27, 数据为 0xFF
AT+DRVI2CWRBYTES=0,0x34,2,0x27FF
```

3.10.10 AT+DRVSPICONFGPIO: 配置 SPI GPIO

设置命令

命令:

```
AT+DRVSPICONFGPIO=<mosi>,<miso>,<sclk>,<cs>
```

响应:

OK

参数

- **<mosi>**: 主出从入信号对应的 GPIO 管脚。
- **<miso>**: 主入从出信号对应 GPIO 管脚, 若不使用, 置位 -1。
- **<sclk>**: SPI 时钟信号对应的 GPIO 管脚。
- **<cs>**: 选择从机的信号对应 GPIO 管脚, 若不使用, 置位 -1。

3.10.11 AT+DRVSPIINIT: 初始化 SPI 主机驱动**设置命令****命令:**

```
AT+DRVSPIINIT=<clock>,<mode>,<cmd_bit>,<addr_bit>,<dma_chan>[,bits_msb]
```

响应:

OK

参数

- **<clock>**: 时钟速度, 分频数为 80 MHz, 单位: Hz, 最大值: 40 MHz。
- **<mode>**: SPI 模式, 范围: 0 ~ 3。
- **<cmd_bit>**: 命令阶段的默认位数, 范围: 0 ~ 16。
- **<addr_bit>**: 地址阶段的默认位数, 范围: 0 ~ 64。
- **<dma_chan>**: 通道 1 或 2, 不需要 DMA 时也可 0。
- **<bits_msb>**: SPI 数据格式:
 - bit 0:
 - * 0: 先传输 MSB (默认);
 - * 1: 先传输 LSB。
 - bit 1:
 - * 0: 先接收 MSB (默认);
 - * 1: 先接收 LSB。

说明

- 请在 SPI 初始化前配置 SPI GPIO。

示例

```
AT+DRVSPIINIT=102400,0,0,0,0,3 // SPI 时钟: 100_
↪kHz; 模式: 0; 命令阶段和地址阶段默认位数均为 0; 不使用 DMA; 先传输和接收 LSB
OK
AT+DRVSPIINIT=0 // 删除 SPI 驱动
OK
```

3.10.12 AT+DRVSPIRD: 读取 SPI 数据

设置命令

命令:

```
AT+DRVSPIRD=<data_len>[,<cmd>,<cmd_len>][,<addr>,<addr_len>]
```

响应:

```
+DRVSPIRD:<read data>  
OK
```

参数

- **<data_len>**: 待读取的 SPI 数据长度, 范围: 1 ~ 4092 字节。
- **<cmd>**: 命令数据, 数据长度由 **<cmd_len>** 参数设定。
- **<cmd_len>**: 本次传输中的命令长度, 范围: 0 ~ 2 字节。
- **<addr>**: 命令地址, 地址长度由 **<addr_len>** 参数设定。
- **<addr_len>**: 本次传输中地址长度, 范围: 0 ~ 4 字节。

说明

- 若不使用 DMA, **<data_len>** 参数每次能够设定的最大值为 64 字节。

示例

```
AT+DRVSPIRD=2 // 读取 2 字节数据  
+DRVI2CREAD:ffff  
OK  
  
AT+DRVSPIRD=2,0x03,1,0x001000,3 // 读取 2 字节数据, <cmd> 为 0x03, <cmd_len> 为 1,  
↪ 字节, <addr> 为 0x1000, <addr_len> 为 3 字节  
+DRVI2CREAD:ffff  
OK
```

3.10.13 AT+DRVSPIWR: 写入 SPI 数据

设置命令

命令:

```
AT+DRVSPIWR=<data_len>[,<cmd>,<cmd_len>][,<addr>,<addr_len>]
```

响应:

当 **<data_len>** 参数值大于 0, AT 返回:

```
OK  
>
```

收到上述响应后, 请输入你想写入的数据, 当数据达到参数指定长度后, 数据传输开始。

若数据传输成功, AT 返回:

OK

当 `<data_len>` 参数值为 0 时，也即 AT 只传输命令和地址，不传输 SPI 数据，此时 AT 返回：

OK

参数

- `<data_len>`：SPI 数据长度，范围：0 ~ 4092。
- `<cmd>`：命令数据，数据长度由 `<cmd_len>` 参数设定。
- `<cmd_len>`：本次传输中的命令长度，范围：0 ~ 2 字节。
- `<addr>`：命令地址，地址长度由 `<addr_len>` 参数设定。
- `<addr_len>`：本次传输中地址长度，范围：0 ~ 4 字节。

说明

- 若不使用 DMA，`<data_len>` 参数每次能够设定的最大值为 64 字节。

示例

```
AT+DRVSPIWR=2 // 写入 2 字节数据
OK
> // 开始接收串行数据
OK

AT+DRVSPIWR=0,0x03,1,0x001000,3 // 写入 0 字节数据，<cmd> 为 0x03，<cmd_len> 为 1 字节，<addr> 为 0x1000，<addr_len> 为 3 字节
OK
```

3.11 Web 服务器 AT 命令

- [介绍](#)
- `AT+WEBSERVER`：启用或禁用通过 Web 服务器配置 Wi-Fi 连接

3.11.1 介绍

重要：默认的 AT 固件不支持此页面下的 AT 命令。如果你需要 ESP32-C6 支持 Web 服务器命令，请自行编译 [ESP-AT 工程](#)，在第五步配置工程里选择：

- 启用 Component config>AT>AT Web Server command support
-

3.11.2 AT+WEBSERVER：启用或禁用通过 Web 服务器配置 Wi-Fi 连接

设置命令

命令：

```
AT+WEBSERVER=<enable>,<server_port>,<connection_timeout>
```

响应:

```
OK
```

参数

- **<enable>**: 启用或禁用 Web 服务器。
 - 0: 禁用 Web 服务器并释放相关资源。
 - 1: 启用 Web 服务器, 你可以通过微信或者浏览器配置 Wi-Fi 连接信息。
- **<server_port>**: Web 服务器端口号。
- **<connection_timeout>**: 每个连接的超时时间。单位: 秒。范围: [21,60]。

说明

- 有两种方法可以提供 Web 服务器所需的 HTML 文件。一种是使用 FAT 文件系统, 此时需要启用 AT FS 命令。另一种是使用嵌入文件来存储 HTML 文件 (默认设置)。
- 默认的 HTML 文件为 [index.html](#)。如果需要自定义 HTML 文件的显示格式或显示文字, 则你直接修改该文件即可; 如果需要自定义 HTML 文件的其它内容 (例如: 增加一个字段), 则你需要对应修改源码文件 [at_web_server_cmd.c](#)。
- 请确保开放的 socket 的最大数目不能小于 12, 你可以在 menuconfig 中设置此项 `./build.py menuconfig>Component config>LWIP>Max number of open sockets`, 然后重新编译工程 (参考文档[本地编译 ESP-AT 工程](#))。
- AT 固件默认不支持 Web 服务器 AT 命令 (参考文档[see ESP-AT 固件差异](#)), 但你可以在 menuconfig 中设置支持 Web 服务器 AT 命令 `./build.py menuconfig>Component config>AT>AT WEB Server command support`, 然后重新编译工程 (参考文档[本地编译 ESP-AT 工程](#))。
- ESP-AT 在 ESP32-C6 系列设备中支持强制门户 (captive portal), 可参考[示例](#)。
- 更多示例可参考文档[Web Server AT 示例](#)。
- 该命令的实现开源, 源码请参考 [at/src/at_web_server_cmd.c](#)。
- 请参考[如何实现 OTA 升级](#) 获取更多 OTA 命令。

示例

```
// 启用 Web 服务器, 端口 80, 每个连接的超时时间 50 秒
AT+WEBSERVER=1,80,50

// 禁用 Web 服务器
AT+WEBSERVER=0
```

3.12 用户 AT 命令

- [介绍](#)
- [AT+USERRAM](#): 操作用户的空闲 RAM
- [AT+USEROTA](#): 根据指定 URL 升级固件
- [AT+USERWKMCFG](#): 设置 AT 唤醒 MCU 的配置
- [AT+USERMCUSLEEP](#): MCU 指示自己睡眠状态
- [AT+USERDOCS](#): 查询固件对应的用户文档链接

3.12.1 介绍

重要：默认的 AT 固件支持此页面下的所有 AT 命令。如果你不需要 ESP32-C6 支持用户命令，请自行[编译 ESP-AT 工程](#)，在第五步配置工程里选择：

- 禁用 Component config>AT>AT user command support
-

3.12.2 AT+USERAM：操作用户的空闲 RAM

查询命令

功能：

查询用户当前可用的空闲 RAM 大小

命令：

```
AT+USERAM?
```

响应：

```
+USERAM:<size>
```

```
OK
```

设置命令

功能：

分配、读、写、擦除、释放在用户 RAM 空间

命令：

```
AT+USERAM=<operation>,<size>[,<offset>]
```

响应：

```
+USERAM:<length>,<data>    // 只有是读操作时，才会有这个回复
```

```
OK
```

参数

- **<operation>**:
 - 0: 释放用户 RAM 空间
 - 1: 分配用户 RAM 空间
 - 2: 向用户 RAM 写数据
 - 3: 从用户 RAM 读数据
 - 4: 清除用户 RAM 上的数据
- **<size>**: 分配/读/写的用户 RAM 大小
- **<offset>**: 读/写 RAM 的偏移量。默认: 0

说明

- 请在执行任何其他操作之前分配用户 RAM 空间。
- 当 <operator> 为 write 时，系统收到此命令后先换行返回 >，此时你可以输入要写的的数据，数据长度应与 <length> 一致。
- 当 <operator> 为 read 时并且长度大于 1024，ESP-AT 会以同样格式多次回复，每次回复最多携带 1024 字节数据，最终以 \r\nOK\r\n 结束。

示例

```
// 分配 1 KB 用户 RAM 空间
AT+USERAM=1,1024

// 向 RAM 空间开始位置写入 500 字节数据
AT+USERAM=2,500

// 从 RAM 空间偏移 100 位置读取 64 字节数据
AT+USERAM=3,64,100

// 释放用户 RAM 空间
AT+USERAM=0
```

3.12.3 AT+USEROTA：根据指定 URL 升级固件

ESP-AT 在运行时，升级到指定 URL 上的新固件。

设置命令

功能：

升级到 URL 指定版本的固件

命令：

```
AT+USEROTA=<url len>
```

响应：

```
OK

>
```

上述响应表示 AT 已准备好接收 URL，此时你可以输入 URL，当 AT 接收到的 URL 长度达到 <url len> 后，返回：

```
Recv <url len> bytes
```

AT 输出上述信息之后，升级过程开始。如果升级完成，返回：

```
OK
```

如果参数错误或者固件升级失败，返回：

```
ERROR
```

参数

- **<url len>**: URL 长度。最大值: 8192 字节

说明

- 你可以从 [GitHub Actions](#) 里下载 所需要的 OTA 固件, 也可以自行编译 [ESP-AT 工程](#) 生成所需要的 OTA 固件。
- OTA 固件为 build/esp-at.bin (非压缩 OTA)。

备注: 压缩 OTA 与非压缩 OTA 区别

- 压缩 OTA (固件后缀 .bin.xz.packed): 采用压缩 OTA 升级方式。芯片重启后, bootloader 会自动校验并解压固件, 然后执行跳转; 若校验失败, 则继续使用旧版固件。解压过程发生在芯片重启后、AT 输出 ready 前, 需要一定时间, 请耐心等待。
- 非压缩 OTA (固件后缀 .bin): 采用非压缩 OTA 升级方式。芯片重启后, bootloader 直接校验固件并执行跳转, 无需解压过程。
- 升级速度取决于网络状况。
- 如果网络条件不佳导致升级失败, AT 将返回 ERROR, 请等待一段时间再试。
- 不建议升级到旧版本。降到旧版本会存在一定的兼容性问题, 甚至无法运行, 如果你坚持要升级到旧版本, 请根据自己的产品自行测试验证功能。
- 建议升级 AT 固件后, 调用 [AT+RESTORE](#) 恢复出厂设置。
- AT+USEROTA 支持 HTTP 和 HTTPS。
- 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)。
- 当 URL 为 HTTPS 时, 不建议 SSL 认证。如果要求 SSL 认证, 你必须自行生成 PKI 文件然后将它们下载到对应的分区中, 之后在 AT+USEROTA 命令的实现代码中加载证书。对于 PKI 文件请参考 [如何更新 PKI 配置](#)。对于 AT+USEROTA 命令, 可参考 ESP-AT 工程提供的示例 [USEROTA](#)。
- 请参考 [如何实现 OTA 升级](#) 获取更多 OTA 命令。

示例

```
AT+USEROTA=36
OK
>
Recv 36 bytes
OK
```

3.12.4 AT+USERWKMCMCFG: 设置 AT 唤醒 MCU 的配置

设置命令

功能:

此命令配置 AT 如何检查 MCU 的唤醒状态, 以及 AT 如何唤醒 MCU。

- 当 MCU 是醒来的状态, AT 将直接向 MCU 发送数据, 不会发送唤醒信号。
- 当 MCU 是睡眠的状态, AT 准备向 MCU 主动发送数据时 (主动发送的数据和 [ESP-AT 消息报告](#) 中定义的不同), AT 会先发送唤醒信号再发送数据。MCU 被唤醒或者超时后会清除唤醒信号。

命令:

```
AT+USERWKMUCCFG=<enable>,<wake mode>,<wake number>,<wake signal>,<delay time>[,
↪<check mcu awake method>]
```

响应:

```
OK
```

参数

- **<enable>**: 启用或禁用唤醒配置。
 - 0: 禁用唤醒 MCU 配置
 - 1: 使能唤醒 MCU 配置
- **<wake mode>**: 唤醒模式。
 - 1: GPIO 唤醒
 - 2: UART 唤醒
- **<wake number>**: 该参数的意义取决于 <wake mode> 的值。
 - 如果 <wake mode> 是 1, <wake number> 代表唤醒管脚 GPIO 编号。用户需要保证配置的唤醒管脚没有用作其它用途, 否则需要用户做兼容性处理。
 - 如果 <wake mode> 是 2, <wake number> 代表唤醒 UART 编号。当前只支持 1, 即支持 UART1 唤醒 MCU。
- **<wake signal>**: 该参数的意义取决于 <wake mode> 的值。
 - 如果 <wake mode> 是 1, <wake signal> 代表唤醒电平。
 - * 0: 低电平
 - * 1: 高电平
 - 如果 <wake mode> 是 2, <wake signal> 代表唤醒字节。范围: [0,255]。
- **<delay time>**: 最大等待时间。单位: 毫秒。范围: [0,60000]。该参数的意义取决于 <wake mode> 的值。
 - 如果 <wake mode> 是 1, 则在 <delay time> 期间内, 将一直保持 <wake signal> 电平。<delay time> 到后, 则反转 <wake signal> 电平。
 - 如果 <wake mode> 是 2, 则立即发送 <wake signal> 字节, 进入等待直到超时。
- **<check mcu awake method>**: AT 检查 MCU 是否处于醒来的状态。
 - bit 0: 是否开启与 `AT+USERMCUSLEEP` 命令的关联。默认开启。即: 收到 `AT+USERMCUSLEEP=0` 命令, 指示 MCU 醒来; 收到 `AT+USERMCUSLEEP=1` 命令, 指示 MCU 睡眠。
 - bit 1: 是否开启与 `AT+SLEEP=0/1/2/3` 命令的关联。默认禁用。即: 收到 `AT+SLEEP=0` 命令, 指示 MCU 醒来; 收到 `AT+SLEEP=1/2/3` 命令, 指示 MCU 睡眠。
 - bit 2: 是否开启 <delay time> 超时后指示 MCU 醒来功能。默认禁用。即: 禁用时, delay time 后, 指示 MCU 睡眠; 使能时, delay time 后, 指示 MCU 醒来。bit 2 需要搭配 bit 0 或 bit 1 一起使用 (单独使用时, 因为默认 MCU 是唤醒状态, 所以每次都不会去唤醒 MCU)。
 - bit 3 (暂未实现): 是否开启 GPIO 指示 MCU 醒来功能。默认不支持。

说明

- 此命令只需要配置一次。
- 每次 AT 向 MCU 主动发送数据前, 会先发送唤醒信号再进入等待, <delay time> 时间到了之后直接发送数据。此超时会降低与 MCU 间的传输效率。
- 如果在 <delay time> 毫秒时间到达之前, AT 收到由 <check mcu awake method> 里的 bit 0 或 bit 1 定义的唤醒事件, 则会立即清除唤醒状态; 否则, AT 将持续等待至 <delay time> 超时, 并在超时后自动清除唤醒状态。

示例

```
// 使能唤醒 MCU 配置。每次 AT 向 MCU 发送数据前, 会先使用 Wi-Fi 模块的 GPIO18_
↪管脚, 高电平唤醒 MCU, 同时保持高电平 10 秒。
AT+USERWKMUCCFG=1,1,18,1,10000,3
```

(续下页)

(接上页)

```
// 禁用唤醒 MCU 配置
AT+USERWKMUCCFG=0
```

3.12.5 AT+USERMCUSLEEP: MCU 指示自己睡眠状态

设置命令

功能:

在`AT+USERWKMUCCFG`命令的 `<check mcu awake method>` bit 0 配置情况下, 此命令才会生效。用于告知 AT 当前 MCU 的睡眠状态。

命令:

```
AT+USERMCUSLEEP=<state>
```

响应:

```
OK
```

参数

- `<state>`:
 - 0: 指示 MCU 醒来。
 - 1: 指示 MCU 睡眠。

示例

```
// MCU 告知 AT 当前 MCU 醒来
AT+USERMCUSLEEP=0
```

3.12.6 AT+USERDOCS: 查询固件对应的用户文档链接

查询命令

功能:

查询当前运行固件对应的中英文用户文档链接。

命令:

```
AT+USERDOCS?
```

响应:

```
+USERDOCS:<"en url">
+USERDOCS:<"cn url">

OK
```

参数

- <" en url" >: 英文文档链接
- <" cn url" >: 中文文档链接

示例

```
AT+USERDOCS?
+USERDOCS:"https://docs.espressif.com/projects/esp-at/en/latest/esp32c6/index.html"
+USERDOCS:"https://docs.espressif.com/projects/esp-at/zh_CN/latest/esp32c6/index.
↪html"

OK
```

强烈建议在使用命令之前先阅读以下内容，了解 AT 命令的一些基本信息。

- [AT 命令分类](#)
- [参数信息保存在 flash 中的 AT 命令](#)
- [AT 消息](#)
- [AT 日志](#)

3.13 AT 命令分类

通用 AT 命令有四种类型：

类型	命令格式	说明
测试命令	AT+< 命令名称 >=?	查询设置命令的内部参数及其取值范围
查询命令	AT+< 命令名称 >?	返回当前参数值
设置命令	AT+< 命令名称 >=<...>	设置用户自定义的参数值，并运行命令
执行命令	AT+< 命令名称 >	运行无用户自定义参数的命令

- 不是每条 AT 命令都具备上述四种类型的命令。
- 命令里输入参数，当前只支持字符串参数和整形数字参数。
- 尖括号 <> 内的参数不可以省略。
- 方括号 [] 内的参数可以省略，省略时使用默认值。例如，运行 **AT+CWJAP** 命令时省略某些参数：

```
AT+CWJAP="ssid","password"
AT+CWJAP="ssid","password","11:22:33:44:55:66"
```

- 当省略的参数后仍有参数要填写时，必须使用 ,，以示分隔，如：

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

- 使用双引号表示字符串参数，如：

```
AT+CWSAP="ESP756290","21030826",1,4
```

- 特殊字符需作转义处理，当前需要转义的字符有 ,、"、\。
 - \\: 转义反斜杠。
 - \,: 转义逗号，分隔参数的逗号无需转义。
 - \": 转义双引号，表示字符串参数的双引号无需转义。
 - \<any>: 转义 <any> 字符，即只使用 <any> 字符，不使用反斜杠。
- 只有 AT 命令中的特殊字符需要转义，其它地方无需转义。例如，AT 命令口打印 > 等待输入数据时，该数据不需要转义。

```
AT+CWJAP="comma\\,backslash\\ssid","1234567890"
AT+MQTTPUB=0,"topic","\\{\\\"sensor\\\":012}\\\"",1,0
```

- AT 命令的默认波特率为 115200。
- 每条 AT 命令的长度不应超过 256 字节。
- AT 命令以新行 (CR-LF) 结束，所以串口工具应设置为“新行模式”。
- AT 命令错误代码的定义请见 [AT API Reference](#):
 - `esp_at_errno_t`
 - `esp_at_para_parse_ret_t`
 - `esp_at_rc_t`

3.14 参数信息保存在 flash 中的 AT 命令

以下 AT 命令的参数更改将始终保存在 flash 的 NVS 区域中，因此重启后，会直接使用。

- **AT+UART_DEF**: AT+UART_DEF=115200,8,1,0,3
- **AT+SAVETRANSLINK**: AT+SAVETRANSLINK=1,"192.168.6.10",1001
- **AT+CWAUTOCONN**: AT+CWAUTOCONN=1

其它一些命令的参数更改是否保存到 flash 可以通过 **AT+SYSTORE** 命令来配置，具体请参见命令的详细说明。

备注： AT 命令里的参数保存，是通过 **NVS** 库实现的。因此，如果命令配置相同的参数值，则不会写入 flash；如果命令配置不同的参数值，flash 也不会被频繁擦除。

3.15 AT 消息

从 ESP-AT 命令端口返回的 ESP-AT 消息有两种类型：ESP-AT 响应（被动）和 ESP-AT 消息报告（主动）。

- **ESP-AT 响应（被动）**
每个输入的 ESP-AT 命令都会返回响应，告诉发送者 ESP-AT 命令的执行结果。响应的最后一条消息必然是 OK 或者 ERROR。

表 3: ESP-AT 响应

AT 响应	说明
OK	AT 命令处理完毕，返回 OK
ERROR	AT 命令错误或执行过程中发生错误
SEND OK	数据已发送到协议栈（针对 AT+CIPSEND 和 AT+CIPSENDEX 命令），但不代表数据已经发到对端
SEND FAIL	向协议栈发送数据时发生错误（针对 AT+CIPSEND 和 AT+CIPSENDEX 命令）
SET OK	URL 已经成功设置（针对 AT+HTTPURLCFG 命令）
+<Command Name>:...	详细描述 AT 命令处理结果

- **ESP-AT 消息报告（主动）**
ESP-AT 会报告系统中重要的状态变化或消息。

表 4: ESP-AT 消息报告

ESP-AT 消息报告	说明
ready	ESP-AT 固件已经准备就绪
busy p...	系统繁忙，正在处理上一条命令，无法处理新的命令

续下页

表 4 - 接上页

ESP-AT 消息报告	说明
ERR CODE:<0x%08x>	不同命令的错误代码
Will force to restart!!!	立即重启模块
smartconfig type:<xxx>	Smartconfig 类型
Smart get wifi info	Smartconfig 已获取 SSID 和 PASSWORD
+SCRD:<length>,<reserved data>	ESP-Touch v2 已获取自定义数据
smartconfig connected wifi	Smartconfig 完成, ESP-AT 已连接到 Wi-Fi
WIFI CONNECTED	Wi-Fi station 接口已连接到 AP
WIFI GOT IP	Wi-Fi station 接口已获取 IPv4 地址
WIFI GOT IPv6 LL	Wi-Fi station 接口已获取 IPv6 链路本地地址
WIFI GOT IPv6 GL	Wi-Fi station 接口已获取 IPv6 全局地址
WIFI DISCONNECT	Wi-Fi station 接口已与 AP 断开连接
+ETH_CONNECTED	以太网接口已连接
+ETH_GOT_IP	以太网接口已获取 IPv4 地址
+ETH_DISCONNECTED	以太网接口已断开
[<conn_id>],CONNECT	ID 为 <conn_id> 的网络连接已建立 (默认情况下, ID 为 0)
[<conn_id>],CLOSED	ID 为 <conn_id> 的网络连接已断开 (默认情况下, ID 为 0)
+ERRNO:<error_code>	命令执行失败时的错误码 - 建立 TCP 连接、UDP 传输、SSL 服务器失败时, 错误码按优先级返回: 首先为套接字的 SO_ERROR; 若 SO_ERROR 为 0, 则返回 LwIP 的 errno; 若 errno 为 0, 则返回 ESP-AT 定义的错误码。 - 建立 SSL 连接失败时, 错误码按优先级返回: 首先为证书相关错误码; 若证书相关错误码为 0, 则返回 mbedTLS 的错误码; 若 mbedTLS 的错误码为 0, 则返回 esp-tls 的错误码; 若 esp-tls 的错误码为 0, 则返回 ESP-AT 定义的错误码。
+LINK_CONN	TCP/UDP/SSL 连接的详细信息
+STA_CONNECTED: <sta_mac>	station 已连接到 ESP-AT 的 Wi-Fi softAP 接口
+DIST_STA_IP: <sta_mac>,<sta_ip>	ESP-AT 的 Wi-Fi softAP 接口给 station 分配 IP 地址
+STA_DISCONNECTED: <sta_mac>	station 与 ESP-AT 的 Wi-Fi softAP 接口的连接断开
>	ESP-AT 正在等待用户输入数据
Recv <xxx> bytes	ESP-AT 从命令端口已接收到 <xxx> 字节

续下页

表 4 - 接上页

ESP-AT 消息报告	说明
+IPD	ESP-AT 在非透传模式下, 已收到来自网络的数据。有以下的消息格式: - 如果 AT+CIPMUX=0, AT+CIPRECVMTYPE=1, 打印: +IPD, <length> - 如果 AT+CIPMUX=1, AT+CIPRECVMTYPE=<link_id>,1, 打印: +IPD,<link_id>,<length> - 如果 AT+CIPMUX=0, AT+CIPRECVMTYPE=0, AT+CIPDINFO=0, 打印: +IPD,<length>:<data> - 如果 AT+CIPMUX=1, AT+CIPRECVMTYPE=<link_id>,0, AT+CIPDINFO=0, 打印: +IPD,<link_id>,<length>:<data> - 如果 AT+CIPMUX=0, AT+CIPRECVMTYPE=0, AT+CIPDINFO=1, 打印: +IPD,<length>,<"remote_ip">,<remote_port>:<data> - 如果 AT+CIPMUX=1, AT+CIPRECVMTYPE=<link_id>,0, AT+CIPDINFO=1, 打印: +IPD,<link_id>,<length>,<"remote_ip">,<remote_port>:<data> 其中的 link_id 为连接 ID, length 为数据长度, remote_ip 为远端 IP 地址, remote_port 为远端端口号, data 为数据。 注意: 当这是个 SSL 连接时, 在被动接收模式下 (AT+CIPRECVMTYPE=1), AT 命令口回复的 length 可能和实际可读的 SSL 数据长度不一致。因为 AT 会优先返回 SSL 层可读的数据长度, 如果 SSL 层可读的数据长度为 0, AT 会返回套接字层可读的数据长度。
透传模式下的数据	ESP-AT 在透传模式下, 已收到来自网络或蓝牙的数据
SEND Canceled	取消在普通传输模式下发送数据
Have <xxx> Connections	已达到服务器的最大连接数
+QUIT	ESP-AT 退出透传模式
NO CERT FOUND	在自定义分区中没有找到有效的设备证书
NO PRVT_KEY FOUND	在自定义分区中没有找到有效的私钥
NO CA FOUND	在自定义分区中没有找到有效的 CA 证书
+TIME_UPDATED	系统时间已更新。只在发送 AT+CIPSNTPCFG 命令后或者掉电重启后, 系统从 SNTP 服务器获取到新的时间, 才会打印此消息
+MQTTCONNECTED	MQTT 已连接到 broker
+MQTTDISCONNECTED	MQTT 与 broker 已断开连接
+MQTTSUBRECV	MQTT 已从 broker 收到数据
+MQTTPUB:FAIL	MQTT 发布数据失败
+MQTTPUB:OK	MQTT 发布数据完成
+BLECONN	Bluetooth LE 连接已建立
+BLEDISCONN	Bluetooth LE 连接已断开
+READ	通过 Bluetooth LE 连接进行读取操作
+WRITE	通过 Bluetooth LE 进行写入操作

续下页

表 4 - 接上页

ESP-AT 消息报告	说明
+NOTIFY	来自 Bluetooth LE 连接的 notification
+INDICATE	来自 Bluetooth LE 连接的 indication
+BLESECNTFYKEY	Bluetooth LE SMP 密钥
+BLESECREQ:<conn_index>	收到来自 Bluetooth LE 连接的加密配对请求
+BLEAUTHCMPL:<conn_index>,<enc_result>	Bluetooth LE SMP 配对完成
+SSID:<ssid>	BluFi 配网模式下, ESP32-C6 收到手机端发送的 SSID
+PASSWORD:<password>	BluFi 配网模式下, ESP32-C6 收到手机端发送的密码
+BLUFIDATA:<len>,<data>	ESP32-C6 收到从手机端发送的 BluFi 用户自定义数据
+WS_DISCONNECTED:<link_id>	连接 ID 为 <link_id> 的 WebSocket 连接已断开
+WS_CONNECTED:<link_id>	连接 ID 为 <link_id> 的 WebSocket 连接已建立
+WS_DATA:<link_id>,<data_len>,<data>	连接 ID 为 <link_id> 的 WebSocket 连接收到数据, 你可以通过 AT+WSDATAFMT 命令来控制接收的数据格式
+WS_CLOSED:<link_id>	连接 ID 为 <link_id> 的 WebSocket 连接已关闭
+BLESCANDONE	扫描结束
+BLESECKEYREQ:<conn_index>	对端已经接受配对请求, ESP 设备可以输入密钥了
+BLE_TT_MODE	进入 BLE 透传模式
+WRITE:<conn_index>	在透传模式下, 使能 notify 或者 indicate 时, 打印以 “+WRITE:” 开头的提示消息

3.16 AT 日志

通过[硬件连接](#)中输出日志功能对应的管脚, ESP-AT 会输出丰富的日志信息, 帮助用户了解 ESP-AT 的运行状态。你也可以[启用不同的日志配置](#), 以查看更多详细的日志信息。下面列出了部分常见的日志信息和对应说明:

表 5: ESP-AT 日志

AT 日志	说明
at-common: write dlen:0	AT 命令口的 RX 收到非预期的数据，通常这些数据里含有多个 \0 字符。如果使用 UART 和 MCU 通信，请检查： • ESP32-C6 的 UART RX 引脚是否正确连接到 MCU 的 TX 引脚 • MCU 是否发送了非 AT 命令格式的数据 • MCU 是否正确配置了 UART 波特率、数据位、停止位、奇偶校验位、流控 • MCU 的 GND 是否正确连接到 ESP32-C6 的 GND 引脚
at-init: at param mode: <mode>	AT 初始化时的参数模式 • 0: 无参数 • 1: 使用 mfg_nvs 分区里的参数 • 2: 使用 factory_param 分区里的参数
at-init: v4.1.0.0 (<src>)	AT 固件的版本号和对对应固件生成的来源
at-wifi: negotiated phy mode: <phy_mode>	Wi-Fi Station 连接到 AP 时协商的 PHY 模式 • 0: Low Rate • 1: 11b • 2: 11g • 3: 11a • 4: HT20 • 5: HT40 • 6: HE20 • 7: VHT20
alloc failed, size:<size> caps:<caps>	内存分配失败，请求大小为 <size>，能力为 <caps>，请考虑分时复用功能或减少内存占用： • 在 <code>python build.py menuconfig</code> > Component config > AT 路径下禁用不需要的 AT 功能 • 参考 内存优化 文档中的优化方法
at-wifi: wifi disconnected, rc:<reason_code>	Wi-Fi station 断开连接的 原因代码
at-wifi: wifi reconnect.(<index>/<total_count>)	Wi-Fi station 正在重连到 AP
at-net: unknown host: <hostname>	无法解析主机名 <hostname>，请检查 DNS 设置和网络连接
at-net: send failed, s:<send_len> r:<ret_len>	发送数据失败，发送长度为 <send_len>，实际返回长度为 <ret_len>。请检查网络连接
at-net: link is changed	AT 作为服务器时，在数据发送过程中，客户端连接断开后，新的客户端连接到服务器
at-net: link:<link_id> is on netif:<netif_id>	连接 <link_id> 在网络接口 <netif_id> 上建立成功了 • 1: Wi-Fi station 接口 • 2: Wi-Fi softAP 接口 • 3: 以太网接口

Chapter 4

AT 命令示例

4.1 AT 响应消息格式控制示例

- 启用系统消息过滤，实现 HTTP 透传下载功能

4.1.1 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.espressif.com/sites/default/files/home/hardware.png>。

1. 恢复出厂设置。

命令：

```
AT+RESTORE
```

响应：

```
OK
```

2. 设置 Wi-Fi 模式为 station。

命令：

```
AT+CWMODE=1
```

响应：

```
OK
```

3. 连接路由器。

命令：

```
AT+CWJAP="espressif","1234567890"
```

响应：

```
WIFI CONNECTED  
WIFI GOT IP
```

```
OK
```

说明：

- 你输入的 SSID 和密码可能跟上述命令中的不同。请使用你的路由器的 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。

命令：

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

响应：

```
OK
```

```
>
```

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：

```
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。

命令：

```
AT+SYSMMSGFILTERCFG=1,0,7
```

响应：

```
OK
```

```
>
```

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：

```
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

命令：

```
AT+SYSMMSGFILTER=1
```

响应：

```
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

命令：

```
ATE0
```

响应：

```
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）。

命令：

```
AT+HTTPGET="https://www.espressif.com/sites/default/files/home/hardware.png",
↪2048,2048,5000
```

响应:

```
// 此处, MCU 将透传接收到整个 https://www.espressif.com/sites/default/files/
↪home/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

命令:

```
AT+SYMSGFILTERCFG=0
```

响应:

```
OK
```

10. 禁用系统消息过滤

命令:

```
AT+SYMSGFILTER=0
```

响应:

```
OK
```

11. 开启回显

命令:

```
ATE1
```

响应:

```
OK
```

4.2 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。

- 介绍
 - 概述
 - 数据类型
 - 其他重要说明
- 出厂参数配置
 - 模组名称操作
 - *UART* 配置操作
 - 最大发射功率操作
 - 国家代码操作
 - 系统存储模式操作
 - 添加自定义键值对
- *PKI* 配置
 - *SSL* 客户端证书
 - *SSL* 服务器证书
- *GATTS* 配置
- 自定义数据存储

4.2.1 介绍

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 客户端或服务端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：

命令：

```
AT+SYSMFG?
```

响应：

```
+SYSMFG:"factory_param"
+SYSMFG:"client_cert"
+SYSMFG:"client_key"
+SYSMFG:"client_ca"
+SYSMFG:"server_cert"
+SYSMFG:"server_key"
+SYSMFG:"server_ca"
...
+SYSMFG:"mqtt_cert"
+SYSMFG:"mqtt_key"
+SYSMFG:"mqtt_ca"

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。

数据类型

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）

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 > 并等待你发送 <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**：你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 命令允许你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](#)。

4.2.2 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。ESP-AT 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+CWOUNTRY 。	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： AT+SYSSSTORE 。	系统存储模式操作

除上表列出的预定义键名外，你也可以在 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。

查询所有出厂参数

命令：

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

响应：

```
+SYSMFG:"factory_param","version",2
+SYSMFG:"factory_param","module_name",7
+SYSMFG:"factory_param","max_tx_power",2
+SYSMFG:"factory_param","country_code",7
+SYSMFG:"factory_param","start_channel",1
+SYSMFG:"factory_param","channel_num",1
+SYSMFG:"factory_param","uart_port",2
```

(续下页)

(接上页)

```
+SYSMFG:"factory_param","uart_baudrate",6
+SYSMFG:"factory_param","uart_tx_pin",6
+SYSMFG:"factory_param","uart_rx_pin",6
+SYSMFG:"factory_param","uart_cts_pin",6
+SYSMFG:"factory_param","uart_rts_pin",6
+SYSMFG:"factory_param","sys_store",1
```

OK

模组名称操作

1. 读取模组名称:

命令:

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

响应:

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

OK

2. 写入新的模组名称:

命令:

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

响应:

OK

>

返回 > 后, 发送模组名称字符串。例如, 发送 “MY-MODULE-01”。

数据发送后的响应:

OK

3. 验证配置是否生效:

命令:

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

响应:

```
+SYSMFG:"factory_param","module_name",7,12,MY-MODULE-01
```

OK

模组名称已成功更新为 “MY-MODULE-01”。

UART 配置操作

1. 读取 UART 端口:

命令:

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

响应:

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

2. 读取 UART 波特率:

命令:

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

响应:

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

3. 更新 UART TX 引脚:

命令:

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

响应:

```
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。

最大发射功率操作

1. 读取最大发射功率:

命令:

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

响应:

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

2. 更新最大发射功率:

命令:

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

响应:

```
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。

- 重启设备使配置生效:

命令:

```
AT+RST
```

响应:

```
ready
```

- 验证配置是否生效:

命令:

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

响应:

```
+SYSMF:"factory_param","max_tx_power",2,2,60
```

```
OK
```

最大发射功率已成功更新。

- 使用 AT+RFPOWER 命令进一步验证:

命令:

```
AT+RFPOWER?
```

响应:

```
+RFPOWER:60
```

```
OK
```

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

国家代码操作

- 读取国家代码:

命令:

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

响应:

```
+SYSMF:"factory_param","country_code",7,2,CN
```

```
OK
```

- 更新国家代码:

命令:

```
AT+SYSMF=2,"factory_param","country_code",7,2
```

响应:

```
OK
```

```
>
```

收到 > 后，发送国家代码（例如 “US”、“JP”、“EU”）。

数据传输后的响应：

```
OK
```

3. 重启设备使配置生效：

命令：

```
AT+RST
```

响应：

```
ready
```

4. 验证配置是否生效：

命令：

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

响应：

```
+SYSFMFG:"factory_param","country_code",7,2,US
```

```
OK
```

国家代码已成功更新为 “US”。

5. 更新信道数量为 11（美国标准）：

命令：

```
AT+SYSFMFG=2,"factory_param","channel_num",1,11
```

响应：

```
OK
```

6. 重启设备使配置生效：

命令：

```
AT+RST
```

响应：

```
ready
```

7. 使用 AT+CWCOUNTRY 命令验证国家码和信道配置：

命令：

```
AT+CWCOUNTRY?
```

响应：

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

```
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。

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：

命令：

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

响应：

```
+SYSMFG:"factory_param","sys_store",1,1,1
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)

2. 更改系统存储模式为不自动存储：

命令：

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

响应：

```
OK
```

3. 重启设备使配置生效：

命令：

```
AT+RST
```

响应：

```
ready
```

4. 验证配置是否生效：

命令：

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

响应：

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

5. 使用 AT+SYSSTORE 命令进一步验证：

命令：

```
AT+SYSSTORE?
```

响应：

```
+SYSSTORE:0
```

```
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。

添加自定义键值对

除了预定义的出厂参数外，你还可以在 `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：

命令：

```
AT+SYSMFG=2,"factory_param","device_sn",7,16
```

响应：

```
OK
```

```
>
```

返回 > 后，发送设备序列号（例如 “SN202601050001”）。

数据发送后的响应：

```
OK
```

2. 读取自定义键值对：

命令：

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

响应：

```
+SYSMFG:"factory_param","device_sn",7,16,SN202601050001
```

```
OK
```

3. 添加数值类型的自定义键值对：

命令：

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

响应：

```
OK
```

这将存储硬件版本号 1001。

4. 查询所有键值对（包括自定义的）：

命令:

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

响应:

```
+SYSMFG:"factory_param","version",2
+SYSMFG:"factory_param","module_name",7
...
+SYSMFG:"factory_param","device_sn",7
+SYSMFG:"factory_param","hw_version",6

OK
```

5. 删除自定义键值对:

命令:

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

响应:

```
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
-

4.2.3 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。ESP-AT 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_cer - client_key - client_ca	- client_cert.0, client_cert.1 - client_key.0, client_key.1 - client_ca.0, 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_cer - server_key - server_ca	- server_cert - server_key - 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 - mqtt_key - mqtt_ca	- mqtt_cert - mqtt_key - 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 - https_key - https_ca	- https_cert.0, https_cert.1 - https_key.0, https_key.1 - https_ca.0, 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。

备注：你可以通过 AT+SYSMFG? 命令查询当前固件支持的所有命名空间，以确认你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。

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）。

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：

命令：

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

响应：

```
+SYSMFG:"client_cert","client_cert.0",8
+SYSMFG:"client_cert","client_cert.1",8
OK
```

同样，你可以查询 client_key 和 client_ca 命名空间：

```
AT+SYSMFG=1,"client_key"
AT+SYSMFG=1,"client_ca"
```

2. 读取特定的证书或密钥：

命令：

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

响应：

```
+SYSMFG:"client_cert","client_cert.0",8,1164,<value>
OK
```

配置证书

1. 写入客户端证书：

命令：

```
AT+SYSMFG=2,"client_cert","client_cert.0",8,1164
```

响应：

```
OK
>
```

返回 > 后，发送证书数据（1164 字节）。数据发送完成后，系统返回：

```
OK
```

2. 写入客户端密钥:

命令:

```
AT+SYSMF=2,"client_key","client_key.0",8,<key_length>
```

响应:

```
OK
```

```
>
```

返回 > 后, 发送密钥数据。数据发送完成后, 系统返回:

```
OK
```

3. 写入 CA 证书:

命令:

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

响应:

```
OK
```

```
>
```

返回 > 后, 发送 CA 证书数据。数据发送完成后, 系统返回:

```
OK
```

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, 你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:

命令:

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

响应:

```
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:

命令:

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

响应:

```
OK
```

4. 建立到服务器的 SSL 连接:

命令:

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

响应:

```
CONNECT
```

```
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。

擦除证书 擦除特定的证书:

命令:

```
AT+SYSMFG=0, "client_cert", "client_cert.0"
```

响应:

```
OK
```

擦除命名空间中的所有键值对:

命令:

```
AT+SYSMFG=0, "client_cert"
```

响应:

```
OK
```

你可以使用相同的命令擦除 client_key 和 client_ca 数据。

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）。

查询证书信息

1. 读取服务器证书命名空间中的所有键:

命令:

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

响应:

```
+SYSMFG:"server_cert", "server_cert.0", 8  
+SYSMFG:"server_cert", "server_cert.1", 8
```

```
OK
```

同样, 你可以查询 server_key 和 server_ca 命名空间:

```
AT+SYSMFG=1, "server_key"  
AT+SYSMFG=1, "server_ca"
```

2. 读取特定的服务器证书或密钥:

命令:

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

响应:

```
+SYSMF:"server_cert","server_cert.0",8,<length>,<value>
```

```
OK
```

命令:

```
AT+SYSMF=1,"server_key","server_key.0"
```

响应:

```
+SYSMF:"server_key","server_key.0",8,<length>,<value>
```

```
OK
```

配置证书

1. 写入服务器证书:

命令:

```
AT+SYSMF=2,"server_cert","server_cert.0",8,<cert_length>
```

响应:

```
OK
```

```
>
```

返回 > 后，发送证书数据。

数据传输后的响应:

```
OK
```

2. 写入服务器密钥:

命令:

```
AT+SYSMF=2,"server_key","server_key.0",8,<key_length>
```

响应:

```
OK
```

```
>
```

返回 > 后，发送密钥数据。

数据传输后的响应:

```
OK
```

3. 写入 CA 证书:

命令:

```
AT+SYSMF=2,"server_ca","server_ca.0",8,<ca_length>
```

响应:

```
OK
```

```
>
```

返回 > 后, 发送 CA 证书数据。

数据传输后的响应:

```
OK
```

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, 你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:

命令:

```
AT+CIPMUX=1
```

响应:

```
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）:

命令:

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

响应:

```
OK
```

- 如果需要双向认证（服务器和客户端相互验证证书）:

命令:

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

响应:

```
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。

4. 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。

擦除证书 擦除特定的服务器证书：

命令：

```
AT+SYSMF0=0,"server_cert","server_cert.0"
```

响应：

```
OK
```

擦除命名空间中的所有键：

命令：

```
AT+SYSMF0=0,"server_cert"
```

响应：

```
OK
```

你可以使用相同的命令擦除 server_key 和 server_ca 数据。

4.2.4 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, ..., 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+BLEINIT

重要：

- 使用 AT+SYSMF0 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+SYSMF0 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。

查询配置信息

读取配置项

- 读取 ble_data 命名空间中的所有配置项：

命令：

```
AT+SYSMF0=1,"ble_data"
```

响应：

```
+SYSMSG:"ble_data","cfg0",7
+SYSMSG:"ble_data","cfg1",7
+SYSMSG:"ble_data","cfg2",7
+SYSMSG:"ble_data","cfg3",7
...
+SYSMSG:"ble_data","cfg30",7

OK
```

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

命令：

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

响应：

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

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）

配置服务

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：

命令：

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

响应：

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

OK
```

当前 UUID 为 0xC300。

2. 写入 UUID 为 0xC500 的新配置：

命令：

```
AT+SYSMSG=2,"ble_data","cfg2",7,23
```

响应：

```
OK

>
```

发送数据:

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

响应:

```
OK
```

3. 查询配置项验证 UUID 是否修改成功:

命令:

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

响应:

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

```
OK
```

UUID 已成功更改为 0xC500。

4. 重新初始化低功耗蓝牙并创建服务以应用更改:

命令:

```
AT+BLEINIT=2
```

响应:

```
OK
```

5. 查询服务以验证:

命令:

```
AT+BLEGATTSSRV?
```

响应:

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

```
OK
```

6. 查询特征以验证 UUID 更改:

命令:

```
AT+BLEGATTCHAR?
```

响应:

```
+BLEGATTCHAR:"char",1,1,0xC500  
+BLEGATTCHAR:"char",1,2,0xC301  
+BLEGATTCHAR:"char",1,3,0xC302  
+BLEGATTCHAR:"char",1,4,0xC303  
+BLEGATTCHAR:"char",1,5,0xC304  
+BLEGATTCHAR:"char",1,6,0xC305  
+BLEGATTCHAR:"char",1,7,0xC306  
+BLEGATTCHAR:"char",1,8,0xC307  
+BLEGATTCHAR:"char",2,1,0xC400  
+BLEGATTCHAR:"char",2,2,0xC401
```

```
OK
```


第一个特征现在显示 UUID 为 0xC500 而不是 0xC300。

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。假设我们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：

命令：

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

响应示例：

```
+SYSMFG:"ble_data","cfg0",7
+SYSMFG:"ble_data","cfg1",7
...
+SYSMFG:"ble_data","cfg30",7
OK
```

假设当前最后一个配置项是 cfg30，我们将从 cfg31 开始添加新服务。

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

命令：

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

响应：

```
OK
```

```
>
```

发送数据：

```
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

响应：

```
OK
```

3. 添加特征声明 (Characteristic Declaration)：

命令：

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

响应：

```
OK
```

```
>
```

发送数据:

```
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

响应:

```
OK
```

4. 添加特征值 (Characteristic Value):

命令:

```
AT+SYSMFG=2,"ble_data","cfg33",7,31
```

响应:

```
OK
```

```
>
```

发送数据:

```
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)

响应:

```
OK
```

5. 查询新添加的配置项验证:

命令:

```
AT+SYSMFG=1,"ble_data","cfg31"
```

响应:

```
+SYSMFG:"ble_data","cfg31",7,26,31,16,0x2800,0x01,2,2,A100
```

```
OK
```

命令:

```
AT+SYSMFG=1,"ble_data","cfg32"
```

响应示例:

```
+SYSMFG:"ble_data","cfg32",7,24,32,16,0x2803,0x01,3,1,02  
OK
```

命令:

```
AT+SYSMFG=1,"ble_data","cfg33"
```

响应示例:

```
+SYSMFG:"ble_data","cfg33",7,31,33,16,0xB100,0x11,20,8,12345678  
OK
```

6. 重新初始化低功耗蓝牙并创建服务以应用更改:

命令:

```
AT+BLEINIT=2
```

响应:

```
OK
```

7. 查询服务验证新服务已添加:

命令:

```
AT+BLEGATTSSRV?
```

响应示例:

```
+BLEGATTSSRV:1,1,0xA002,1  
+BLEGATTSSRV:1,2,0xA003,1  
+BLEGATTSSRV:1,3,0xA100,1  
OK
```

新添加的服务 UUID 0xA100 已出现在服务列表中。

8. 查询特征验证新特征已添加:

命令:

```
AT+BLEGATTCHAR?
```

响应示例:

```
+BLEGATTCHAR:"char",1,1,0xC300  
+BLEGATTCHAR:"char",1,2,0xC301  
...  
+BLEGATTCHAR:"char",3,1,0xB100  
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+SYSMFGE 写入的配置需重新初始化 BLE（AT+BLEINIT）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](#)

4.2.5 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，你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。

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：

命令：

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

响应：

```
OK
```

```
>
```

返回 > 后，发送你的二进制数据（100 字节）。数据发送完成后，系统返回：

```
OK
```

命名空间 `my_app_ca` 现已创建，其中包含一个键 `my_app_ca.0`，存储了你的数据。

2. 查询自定义命名空间：

命令：

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

响应：

```
+SYSMFGE:"my_app_ca","my_app_ca.0",8
```

```
OK
```

3. 从自定义命名空间读取数据：

命令：

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

响应:

```
+SYSMFG:"my_app_ca","my_app_ca.0",8,100,<data>  
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。

4. 向同一命名空间添加更多键:

命令:

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

响应:

```
OK  
>
```

发送数据:

```
MyCustomConfigValue1
```

响应:

```
OK
```

5. 查询命名空间中的所有键:

命令:

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

响应:

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

6. 创建另一个命名空间存储应用配置参数:

如果你需要存储配置参数（而非证书数据），建议创建单独的命名空间以保持语义清晰。

命令:

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

响应:

```
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。

7. 读取配置参数:

命令:

```
AT+SYSMFG=1,"my_app_config","timeout"
```

响应:

```
+SYSFMFG:"my_app_config","timeout",6,4,5000
```

```
OK
```

8. 擦除特定的键:

命令:

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

响应:

```
OK
```

9. 擦除整个命名空间:

命令:

```
AT+SYSFMFG=0,"my_app_ca"
```

响应:

```
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+SYSFMFG? 查询, 在命名空间列表中看到它

4.3 TCP-IP AT 示例

本文档主要介绍在 ESP32-C6 设备上运行 [TCP/IP AT 命令](#) 命令的详细示例。

- [ESP32-C6 设备作为 TCP 客户端建立单连接](#)
- [ESP32-C6 设备作为 TCP 服务器建立多连接](#)
- [远端 IP 地址和端口固定的 UDP 通信](#)
- [远端 IP 地址和端口可变的 UDP 通信](#)
- [ESP32-C6 设备作为 SSL 客户端建立单连接](#)
- [ESP32-C6 设备作为 SSL 服务器建立多连接](#)
- [ESP32-C6 设备作为 SSL 客户端建立双向认证单连接](#)
- [ESP32-C6 设备作为 SSL 服务器建立双向认证多连接](#)
- [ESP32-C6 设备作为 TCP 客户端, 建立单连接, 实现 Network 透传](#)
- [ESP32-C6 设备作为 TCP 服务器, 实现 Network 透传](#)
- [ESP32-C6 设备作为 softAP 在 UDP 传输中实现 Network 透传](#)
- [ESP32-C6 设备获取被动接收模式下的套接字数据](#)
- [ESP32-C6 设备开启 mDNS 服务, PC 通过域名连接到设备的 TCP 服务器](#)

4.3.1 ESP32-C6 设备作为 TCP 客户端建立单连接

1. 设置 Wi-Fi 模式为 station。

命令：

```
AT+CWMODE=1
```

响应：

```
OK
```

2. 连接到路由器。

命令：

```
AT+CWJAP="espressif","1234567890"
```

响应：

```
WIFI CONNECTED
WIFI GOT IP
OK
```

说明：

- 你输入的 SSID 和密码可能跟上述命令中的不同。请使用你的路由器的 SSID 和密码。

3. 查询 ESP32-C6 设备 IP 地址。

命令：

```
AT+CIPSTA?
```

响应：

```
+CIPSTA:ip:"192.168.3.112"
+CIPSTA:gateway:"192.168.3.1"
+CIPSTA:netmask:"255.255.255.0"
OK
```

说明：

- 你的查询结果可能与上述响应中的不同。

4. PC 与 ESP32-C6 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. ESP32-C6 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。

命令：

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

响应：

```
CONNECT
OK
```

6. 发送 4 字节数据。

命令：

```
AT+CIPSEND=4
```

响应：

```
OK
>
```

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。

```
Recv 4 bytes
```

```
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：

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

4.3.2 ESP32-C6 设备作为 TCP 服务器建立多连接

当 ESP32-C6 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 ESP32-C6 设备。

以下是 ESP32-C6 设备作为 softAP 建立 TCP 服务器的示例；如果是 ESP32-C6 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。

命令：

```
AT+CWMODE=2
```

响应：

```
OK
```

2. 使能多连接。

命令：

```
AT+CIPMUX=1
```

响应：

```
OK
```

3. 设置 softAP。

命令：

```
AT+CWSAP="ESP32_softAP","1234567890",5,3
```

响应：

```
OK
```

4. 查询 softAP 信息。

命令：

```
AT+CIPAP?
```

响应：

```
AT+CIPAP?
+CIPAP:ip:"192.168.4.1"
+CIPAP:gateway:"192.168.4.1"
+CIPAP:netmask:"255.255.255.0"
```

```
OK
```

说明：

- 你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。

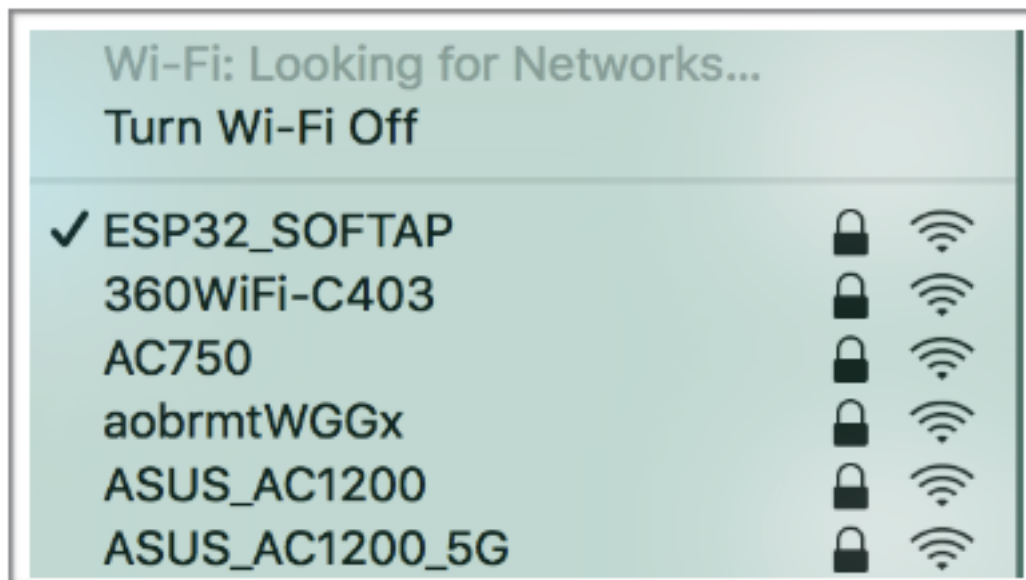
命令：


```
AT+CIPSERVER=1
```

响应：

```
OK
```

6. PC 连接到 ESP32-C6 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 ESP32-C6 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。

命令：

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

响应：

```
OK
```

```
>
```

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。

```
Recv 4 bytes
```

```
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：

```
+IPD,0,4:test
```

10. 关闭 TCP 连接。

命令：

```
AT+CIPCLOSE=0
```

响应：

```
0,CLOSED
```

```
OK
```

4.3.3 远端 IP 地址和端口固定的 UDP 通信

1. 设置 Wi-Fi 模式为 station。

命令：

```
AT+CWMODE=1
```

响应：

```
OK
```

2. 连接到路由器。

命令：

```
AT+CWJAP="espressif","1234567890"
```

响应：

```
WIFI CONNECTED
WIFI GOT IP
OK
```

说明：

- 你输入的 SSID 和密码可能跟上述命令中的不同。请使用你的路由器的 SSID 和密码。

3. 查询 ESP32-C6 设备 IP 地址。

命令：

```
AT+CIPSTA?
```

响应：

```
+CIPSTA:ip:"192.168.3.112"
+CIPSTA:gateway:"192.168.3.1"
+CIPSTA:netmask:"255.255.255.0"
OK
```

说明：

- 你的查询结果可能与上述响应中的不同。

4. PC 与 ESP32-C6 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。

命令：

```
AT+CIPMUX=1
```

响应：

```
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+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。

命令：

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

响应：

```
4,CONNECT
```

```
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 ESP32-C6 设备的 UDP 本地端口，你可自行设置，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 ESP32-C6 设备端口 1112 发送数据，ESP32-C6 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。

命令：

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

响应：

```
OK
```

```
>
```

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。

```
Recv 7 bytes
```

```
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：

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

9. 关闭网络连接 ID 为 4 的 UDP 连接。

命令：

```
AT+CIPCLOSE=4
```

响应：

```
4,CLOSED
```

```
OK
```

4.3.4 远端 IP 地址和端口可变的 UDP 通信

1. 设置 Wi-Fi 模式为 station。

命令：

```
AT+CWMODE=1
```

响应：

```
OK
```

2. 连接到路由器。

命令：

```
AT+CWJAP="espressif","1234567890"
```

响应:

```
WIFI CONNECTED
WIFI GOT IP
OK
```

说明:

- 你输入的 SSID 和密码可能跟上述命令中的不同。请使用你的路由器的 SSID 和密码。

3. 查询 ESP32-C6 设备 IP 地址。

命令:

```
AT+CIPSTA?
```

响应:

```
+CIPSTA:ip:"192.168.3.112"
+CIPSTA:gateway:"192.168.3.1"
+CIPSTA:netmask:"255.255.255.0"
OK
```

说明:

- 你的查询结果可能与上述响应中的不同。

4. PC 与 ESP32-C6 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。

命令:

```
AT+CIPMUX=0
```

响应:

```
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。

命令:

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

响应:

```
CONNECT
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 ESP32-C6 设备的 UDP 本地端口, 你可自行设置, 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 ESP32-C6 设备 UDP 通信的远端 IP 地址和端口。

7. 发送 4 字节数据。

命令:

```
AT+CIPSEND=4
```

响应:

```
OK
>
```

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。

```
Recv 4 bytes
```

```
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。

命令：

```
AT+CIPSEND=4,"192.168.3.103",1000
```

响应：

```
OK
```

```
>
```

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。

```
Recv 4 bytes
```

```
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：

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

10. 关闭 UDP 连接。

命令：

```
AT+CIPCLOSE
```

响应：

```
CLOSED
```

```
OK
```

4.3.5 ESP32-C6 设备作为 SSL 客户端建立单连接

1. 设置 Wi-Fi 模式为 station。

命令：

```
AT+CWMODE=1
```

响应：

```
OK
```

2. 连接到路由器。

命令：

```
AT+CWJAP="espressif","1234567890"
```

响应：

```
WIFI CONNECTED
```

```
WIFI GOT IP
```

(续下页)

(接上页)

OK

说明:

- 你输入的 SSID 和密码可能跟上述命令中的不同。请使用你的路由器的 SSID 和密码。

3. 查询 ESP32-C6 设备 IP 地址。

命令:

AT+CIPSTA?

响应:

```
+CIPSTA:ip:"192.168.3.112"
+CIPSTA:gateway:"192.168.3.1"
+CIPSTA:netmask:"255.255.255.0"
```

OK

说明:

- 你的查询结果可能与上述响应中的不同。

4. PC 与 ESP32-C6 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。

命令:

```
openssl s_server -cert /home/esp-at/components/customized_partitions/raw_data/
→server_cert/server_cert.crt -key /home/esp-at/components/customized_
→partitions/raw_data/server_key/server.key -port 8070
```

响应:

ACCEPT

6. ESP32-C6 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。

命令:

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

响应:

CONNECT

OK

7. 发送 4 字节数据。

命令:

AT+CIPSEND=4

响应:

OK

>

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。

Recv 4 bytes

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：

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

4.3.6 ESP32-C6 设备作为 SSL 服务器建立多连接

当 ESP32-C6 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 ESP32-C6 设备。

以下是 ESP32-C6 设备作为 softAP 建立 SSL 服务器的示例；如果是 ESP32-C6 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。

命令：

```
AT+CWMODE=2
```

响应：

```
OK
```

2. 使能多连接。

命令：

```
AT+CIPMUX=1
```

响应：

```
OK
```

3. 配置 ESP32-C6 softAP。

命令：

```
AT+CWSAP="ESP32_softAP","1234567890",5,3
```

响应：

```
OK
```

4. 查询 softAP 信息。

命令：

```
AT+CIPAP?
```

响应：

```
AT+CIPAP?  
+CIPAP:ip:"192.168.4.1"  
+CIPAP:gateway:"192.168.4.1"  
+CIPAP:netmask:"255.255.255.0"
```

```
OK
```

说明：

- 你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。

命令：

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

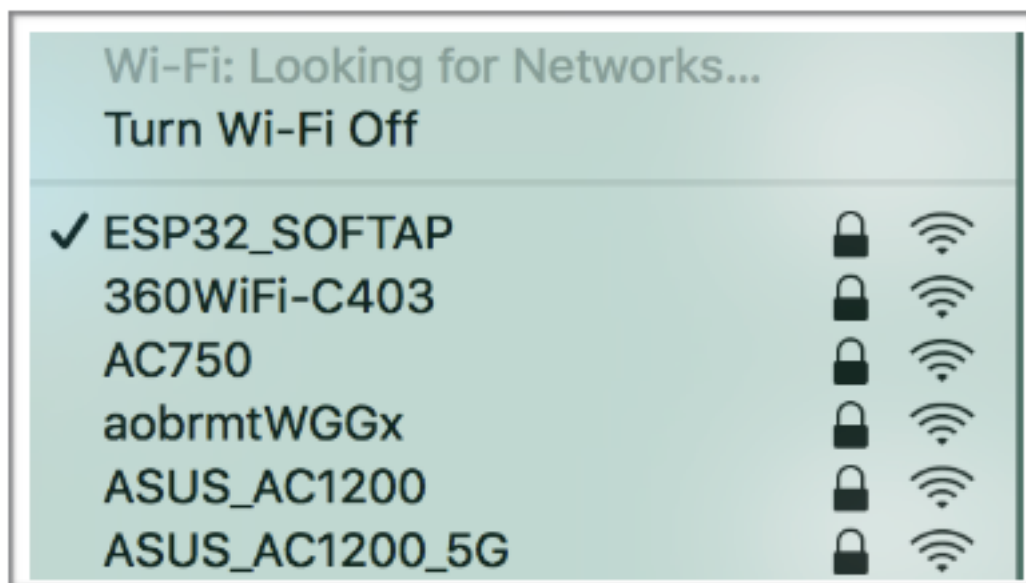
响应：

```
OK
```

6. PC 连接到 ESP32-C6 设备的 softAP。

7. 在 PC 上使用 OpenSSL 命令，创建一个 SSL 客户端，连接到 ESP32-C6 设备创建的 SSL 服务器。

命令：



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

ESP32-C6 设备上的响应：

```
CONNECT
```

8. 发送 4 字节数据到网络连接 ID 为 0 的链路上。

命令：

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

响应：

```
OK
```

```
>
```

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。

```
Recv 4 bytes
```

```
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：

```
+IPD,0,4:test
```

10. 关闭 SSL 连接。

命令：

```
AT+CIPCLOSE=0
```

响应：

```
0,CLOSED
```

```
OK
```


4.3.7 ESP32-C6 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 ESP-AT 中默认的证书，你也可以使用自己的证书：

- 要使用你自己的 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](#)）。
- 要使用你自己的 SSL 服务器证书，请用你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。

命令：

```
AT+CWMODE=1
```

响应：

```
OK
```

2. 连接到路由器。

命令：

```
AT+CWJAP="espressif","1234567890"
```

响应：

```
WIFI CONNECTED
WIFI GOT IP
OK
```

说明：

- 你输入的 SSID 和密码可能跟上述命令中的不同。请使用你的路由器的 SSID 和密码。

3. 设置 SNTP 服务器。

命令：

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

响应：

```
OK
```

说明：

- 你可以根据自己国家的时区设置 SNTP 服务器。

4. 查询 SNTP 时间。

命令：

```
AT+CIPSNTPTIME?
```

响应：

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

说明：

- 你可以查询 SNTP 时间与实时时间是否相符来判断你设置的 SNTP 服务器是否生效。

5. 查询 ESP32-C6 设备 IP 地址。

命令：

```
AT+CIPSTA?
```

响应：

```
+CIPSTA:ip:"192.168.3.112"
+CIPSTA:gateway:"192.168.3.1"
+CIPSTA:netmask:"255.255.255.0"
OK
```

说明：

- 你的查询结果可能与上述响应中的不同。
6. PC 与 ESP32-C6 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。
命令:

```
openssl s_server -CAfile /home/esp-at/components/customized_partitions/raw_
→data/server_ca/server_ca.crt -cert /home/esp-at/components/customized_
→partitions/raw_data/server_cert/server_cert.crt -key /home/esp-at/components/
→customized_partitions/raw_data/server_key/server.key -port 8070 -verify_
→return_error -verify_depth 1 -Verify 1
```

ESP32-C6 设备上的响应:

```
ACCEPT
```

说明:

- 命令中的证书路径可以根据你的证书位置进行调整。
8. ESP32-C6 设备设置 SSL 客户端双向认证配置。
命令:

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

响应:

```
OK
```

9. ESP32-C6 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。
命令:

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

响应:

```
CONNECT
```

```
OK
```

10. 发送 4 字节数据。
命令:

```
AT+CIPSEND=4
```

响应:

```
OK
```

```
>
```

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。

```
Recv 4 bytes
```

```
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:

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

4.3.8 ESP32-C6 设备作为 SSL 服务器建立双向认证多连接

当 ESP32-C6 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 ESP32-C6 设备。

以下是 ESP32-C6 设备作为 station 建立 SSL 服务器的示例；如果是 ESP32-C6 设备作为 softAP，可参考 ESP32-C6 设备作为 SSL 服务器建立多连接示例。

1. 设置 Wi-Fi 模式为 station。

命令：

```
AT+CWMODE=1
```

响应：

```
OK
```

2. 连接到路由器。

命令：

```
AT+CWJAP="espressif","1234567890"
```

响应：

```
WIFI CONNECTED
WIFI GOT IP
OK
```

说明：

- 你输入的 SSID 和密码可能跟上述命令中的不同。请使用你的路由器的 SSID 和密码。

3. 查询 ESP32-C6 设备 IP 地址。

命令：

```
AT+CIPSTA?
```

响应：

```
+CIPSTA:ip:"192.168.3.112"
+CIPSTA:gateway:"192.168.3.1"
+CIPSTA:netmask:"255.255.255.0"
OK
```

说明：

- 你的查询结果可能与上述响应中的不同。

4. 使能多连接。

命令：

```
AT+CIPMUX=1
```

响应：

```
OK
```

5. 建立 SSL 服务器，端口为 8070。

命令：

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

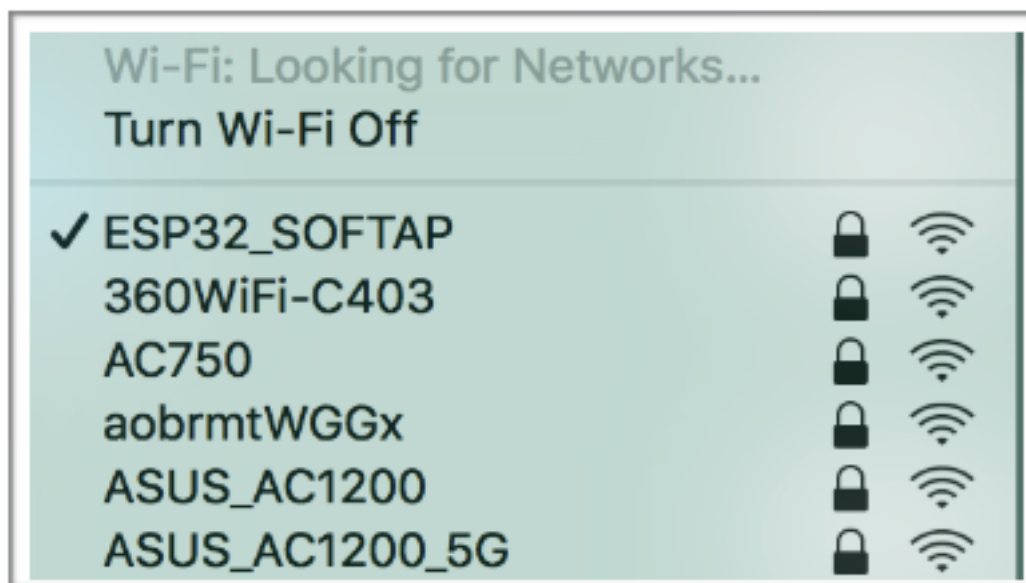
响应：

```
OK
```

6. PC 与 ESP32-C6 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 ESP32-C6 设备创建的 SSL 服务器。

命令：



```
openssl s_client -CAfile /home/esp-at/components/customized_partitions/raw_
→data/client_ca/client_ca_00.crt -cert /home/esp-at/components/customized_
→partitions/raw_data/client_cert/client_cert_00.crt -key /home/esp-at/
→components/customized_partitions/raw_data/client_key/client_key_00.key -host_
→192.168.3.112 -port 8070
```

ESP32-C6 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。

命令：

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

响应：

```
OK
```

```
>
```

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。

```
Recv 4 bytes
```

```
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：

```
+IPD,0,4:test
```

10. 关闭 SSL 连接。

命令：

```
AT+CIPCLOSE=0
```

响应：

```
0,CLOSED
```

```
OK
```

11. 关闭 SSL 服务端。

命令：

```
AT+CIPSERVER=0
```

响应：

```
OK
```

4.3.9 ESP32-C6 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。

命令：

```
AT+CWMODE=1
```

响应：

```
OK
```

2. 连接到路由器。

命令：

```
AT+CWJAP="espressif","1234567890"
```

响应：

```
WIFI CONNECTED  
WIFI GOT IP
```

```
OK
```

说明：

- 你输入的 SSID 和密码可能跟上述命令中的不同。请使用你的路由器的 SSID 和密码。

3. 查询 ESP32-C6 设备 IP 地址。

命令：

```
AT+CIPSTA?
```

响应：

```
+CIPSTA:ip:"192.168.3.112"  
+CIPSTA:gateway:"192.168.3.1"  
+CIPSTA:netmask:"255.255.255.0"
```

```
OK
```

说明：

- 你的查询结果可能与上述响应中的不同。

4. PC 与 ESP32-C6 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. ESP32-C6 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。

命令：

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

响应：

```
CONNECT
```

```
OK
```

6. 进入 Network [透传接收模式](#)。

命令：

```
AT+CIPMODE=1
```

响应：

```
OK
```

7. 进入 Network [透传模式](#) 并发送数据。

命令：

```
AT+CIPSEND
```

响应：

```
OK
```

```
>
```

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。你也可以使用 AT+CIPSEND 命令恢复透传。

9. 退出 Network [透传接收模式](#)。

命令：

```
AT+CIPMODE=0
```

响应：

```
OK
```

10. 关闭 TCP 连接。

命令：

```
AT+CIPCLOSE
```

响应：

```
CLOSED
```

```
OK
```

4.3.10 ESP32-C6 设备作为 TCP 服务器，实现 Network 透传

1. 设置 Wi-Fi 模式为 station。

命令：

```
AT+CWMODE=1
```

响应：

```
OK
```

2. 连接到路由器。

命令：

```
AT+CWJAP="espressif","1234567890"
```

响应:

```
WIFI CONNECTED
WIFI GOT IP
OK
```

说明:

- 你输入的 SSID 和密码可能跟上述命令中的不同。请使用你的路由器的 SSID 和密码。

3. 设置多连接模式。

命令:

```
AT+CIPMUX=1
```

响应:

```
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。

命令:

```
AT+CIPSERVERMAXCONN=1
```

响应:

```
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。

命令:

```
AT+CIPSERVER=1,8080
```

响应:

```
OK
```

说明:

- 设置 TCP 服务器端口为 8080, 你也可以设置为其它端口。

6. 查询 ESP32-C6 设备 IP 地址。

命令:

```
AT+CIPSTA?
```

响应:

```
+CIPSTA:ip:"192.168.3.112"
+CIPSTA:gateway:"192.168.3.1"
+CIPSTA:netmask:"255.255.255.0"
OK
```

说明:

- 你的查询结果可能与上述响应中的不同。

7. PC 连接到 ESP32-C6 TCP 服务器。

PC 与 ESP32-C6 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 ESP32-C6 的 TCP 服务器。地址为 192.168.3.112, 端口为 8080。

AT 响应:

```
0,CONNECT
```

8. 进入 Network [透传接收模式](#)。

命令：

```
AT+CIPMODE=1
```

响应：

```
OK
```

9. 进入 Network [透传模式](#) 并发送数据。

命令：

```
AT+CIPSEND
```

响应：

```
OK
```

```
>
```

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。你也可以使用 AT+CIPSEND 命令恢复透传。

11. 退出 Network [透传接收模式](#)。

命令：

```
AT+CIPMODE=0
```

响应：

```
OK
```

12. 关闭 TCP 连接。

命令：

```
AT+CIPCLOSE
```

响应：

```
CLOSED
```

```
OK
```

4.3.11 ESP32-C6 设备作为 softAP 在 UDP 传输中实现 Network 透传

1. 设置 Wi-Fi 模式为 softAP。

命令：

```
AT+CWMODE=2
```

响应：

```
OK
```

2. 设置 softAP。

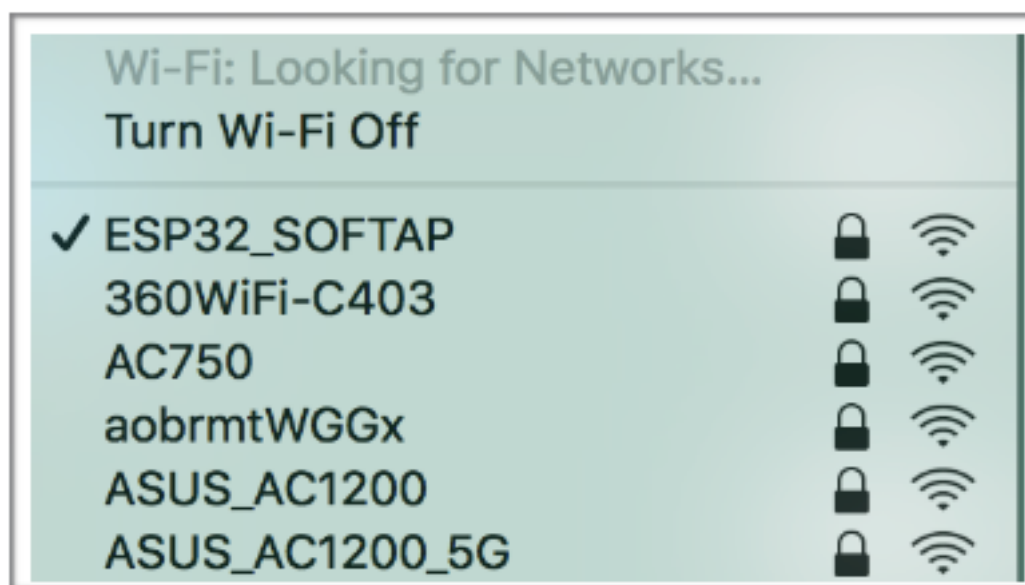
命令：


```
AT+CWSAP="ESP32_softAP","1234567890",5,3
```

响应:

```
OK
```

3. PC 连接到 ESP32-C6 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. ESP32-C6 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。

命令:

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

响应:

```
CONNECT
```

```
OK
```

6. 进入 Network 透传接收模式。

命令:

```
AT+CIPMODE=1
```

响应:

```
OK
```

7. 进入 Network 透传模式 并发送数据。

命令:

```
AT+CIPSEND
```

响应:

```
OK
```

```
>
```

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。你也可以使用 AT+CIPSEND 命令恢复透传。

9. 退出 Network 透传接收模式。

命令：

```
AT+CIPMODE=0
```

响应：

```
OK
```

10. 关闭 TCP 连接。

命令：

```
AT+CIPCLOSE
```

响应：

```
CLOSED
```

```
OK
```

4.3.12 ESP32-C6 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。

命令：

```
AT+CWMODE=1
```

响应：

```
OK
```

2. 连接到路由器。

命令：

```
AT+CWJAP="espressif","1234567890"
```

响应：

```
WIFI CONNECTED
WIFI GOT IP
```

```
OK
```

说明：

- 你输入的 SSID 和密码可能跟上述命令中的不同。请使用你的路由器的 SSID 和密码。

3. 查询 ESP32-C6 设备 IP 地址。

命令：

```
AT+CIPSTA?
```

响应：

```
+CIPSTA:ip:"192.168.3.112"
+CIPSTA:gateway:"192.168.3.1"
+CIPSTA:netmask:"255.255.255.0"
```

(续下页)

(接上页)

OK

说明:

- 你的查询结果可能与上述响应中的不同。

4. PC 与 ESP32-C6 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. ESP32-C6 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。

命令:

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

响应:

CONNECT

OK

6. ESP32-C6 设备设置套接字接收模式为被动模式。

命令:

```
AT+CIPRECVMODE=1
```

响应:

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. ESP32-C6 设备查询被动接收模式下套接字数据的长度。

命令:

```
AT+CIPRECLEN?
```

响应:

+CIPRECLEN:4

OK

9. ESP32-C6 设备获取被动接收模式下的套接字数据。

命令:

```
AT+CIPRECVDATA=4
```

响应:

+CIPRECVDATA:4,test

OK

4.3.13 ESP32-C6 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。

命令:

```
AT+CWMODE=1
```

响应:

```
OK
```

2. 连接到路由器。

命令：

```
AT+CWJAP="espressif","1234567890"
```

响应：

```
WIFI CONNECTED
WIFI GOT IP
```

```
OK
```

说明：

- 你输入的 SSID 和密码可能跟上述命令中的不同。请使用你的路由器的 SSID 和密码。

3. PC 与 ESP32-C6 设备连接同一个路由。

只有在同一个局域网中，PC 才能发现 ESP32-C6 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](#))。

命令：

```
sudo avahi-browse -a -r
```

5. ESP32-C6 设备开启 mDNS 服务。

命令：

```
AT+MDNS=1,"espressif","_printer",35,"my_instance","_tcp",2,"product","my_
→printer","firmware_version","AT-V3.4.1.0"
```

响应：

```
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 ESP32-C6 设备。

PC 端的 avahi-browse 工具会提示：

```
...
+ enx000ec6dd4ebf IPv4 my_instance                               UNIX_
→Printer             local
= enx000ec6dd4ebf IPv4 my_instance                               UNIX_
→Printer             local
  hostname = [espressif.local]
  address  = [192.168.200.90]
  port     = [35]
  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 ESP32-C6 设备的 mDNS 功能是否正常。

7. ESP32-C6 设备使能多连接。

命令：

```
AT+CIPMUX=1
```

响应：

```
OK
```

8. ESP32-C6 设备作为 TCP 服务器，端口为 35。

命令：

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

响应：

```
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 ESP32-C6 设备的 TCP 服务器。

命令：

```
nc espressif.local 35
```

ESP32-C6 设备响应：

```
0,CONNECT
```

说明：

- 连接建立成功后，PC 和 ESP32-C6 设备之间立即可以进行数据传输。

10. ESP32-C6 设备关闭 mDNS 功能。

命令：

```
AT+MDNS=0
```

响应：

```
OK
```

说明：

- 关闭 mDNS 功能能一定程度上减少设备的功耗。

4.4 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 ESP32-C6 设备上运行这些命令的详细示例。

- 简介
- *Bluetooth LE* 客户端读写服务特征值
- *Bluetooth LE* 连接加密
- 两个 ESP32-C6 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
- ESP32-C6 与手机建立 SPP 连接，以及在 *UART-Bluetooth LE* 透传模式下传输数据
- ESP32-C6 和手机之间建立 *Bluetooth LE* 连接并配对

4.4.1 简介

当前，ESP32-C6 AT 固件支持 蓝牙核心规范 5.0 版本。

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。

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，ESP-AT 支持 Bluetooth LE 服务端和 Bluetooth LE 客户端同时存在。
- 不论 ESP-AT 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。

4.4.2 Bluetooth LE 客户端读写服务特征值

以下示例同时使用两块 ESP32-C6 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 ESP32-C6 Bluetooth LE 服务端开头的操作只需要在 ESP32-C6 Bluetooth LE 服务端执行即可，以 ESP32-C6 Bluetooth LE 客户端开头的操作只需要在 ESP32-C6 Bluetooth LE 客户端执行即可。

1. 初始化 Bluetooth LE 功能。

ESP32-C6 Bluetooth LE 服务端:

命令:

```
AT+BLEINIT=2
```

响应:

```
OK
```

ESP32-C6 Bluetooth LE 客户端:

命令:

```
AT+BLEINIT=1
```

响应:

```
OK
```

2. ESP32-C6 Bluetooth LE 服务端获取其 MAC 地址。

命令:

```
AT+BLEADDR?
```

响应:

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

```
OK
```

说明:

- 你查询到的地址可能与上述响应中的不同, 请记住你的地址, 下面的步骤中会用到。

1. ESP32-C6 Bluetooth LE 服务端发现服务特征。

命令:

```
AT+BLEGATTCHAR?
```

响应:

```
+BLEGATTCHAR: "char", 1, 1, 0xC300, 0x02
+BLEGATTCHAR: "desc", 1, 1, 1, 0x2901
+BLEGATTCHAR: "char", 1, 2, 0xC301, 0x02
+BLEGATTCHAR: "desc", 1, 2, 1, 0x2901
+BLEGATTCHAR: "char", 1, 3, 0xC302, 0x08
+BLEGATTCHAR: "desc", 1, 3, 1, 0x2901
+BLEGATTCHAR: "char", 1, 4, 0xC303, 0x04
+BLEGATTCHAR: "desc", 1, 4, 1, 0x2901
+BLEGATTCHAR: "char", 1, 5, 0xC304, 0x08
+BLEGATTCHAR: "char", 1, 6, 0xC305, 0x10
+BLEGATTCHAR: "desc", 1, 6, 1, 0x2902
+BLEGATTCHAR: "char", 1, 7, 0xC306, 0x20
+BLEGATTCHAR: "desc", 1, 7, 1, 0x2902
+BLEGATTCHAR: "char", 1, 8, 0xC307, 0x02
+BLEGATTCHAR: "desc", 1, 8, 1, 0x2901
+BLEGATTCHAR: "char", 2, 1, 0xC400, 0x02
+BLEGATTCHAR: "desc", 2, 1, 1, 0x2901
+BLEGATTCHAR: "char", 2, 2, 0xC401, 0x02
+BLEGATTCHAR: "desc", 2, 2, 1, 0x2901
```

```
OK
```

2. ESP32-C6 Bluetooth LE 服务端开始广播, 之后 ESP32-C6 Bluetooth LE 客户端开始扫描并且持续 3 秒钟。

ESP32-C6 Bluetooth LE 服务端:

命令:

```
AT+BLEADVSTART
```

响应:

```
OK
```

ESP32-C6 Bluetooth LE 客户端:

命令:

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

响应:

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

说明：

- 你的扫描结果可能与上述响应中的不同。

3. 建立 Bluetooth LE 连接。

ESP32-C6 Bluetooth LE 客户端：

命令：

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

响应：

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

OK

说明：

- 输入上述命令时，请使用你的 ESP32-C6 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。

4. ESP32-C6 Bluetooth LE 客户端发现服务。

命令：

```
AT+BLEGATTCPRIMSRV=0
```

响应：

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

OK

说明：

- ESP32-C6 Bluetooth LE 客户端查询服务的结果，比 ESP32-C6 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，ESP32-C6 Bluetooth LE 客户端查询的 <srv_index> 值等于 ESP32-C6 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 ESP32-C6 Bluetooth LE 客户端查询到的 <srv_index> 为 3，如果在 ESP32-C6 Bluetooth LE 服务端通过 [AT+BLEGATTSSRV?](#) 命令查询，则 <srv_index> 为 1。

5. ESP32-C6 Bluetooth LE 客户端发现特征值。

命令：

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

响应：

```
+BLEGATTCCHAR:"char",0,3,1,0xC300,0x02
+BLEGATTCCHAR:"desc",0,3,1,1,0x2901
+BLEGATTCCHAR:"char",0,3,2,0xC301,0x02
+BLEGATTCCHAR:"desc",0,3,2,1,0x2901
+BLEGATTCCHAR:"char",0,3,3,0xC302,0x08
+BLEGATTCCHAR:"desc",0,3,3,1,0x2901
+BLEGATTCCHAR:"char",0,3,4,0xC303,0x04
+BLEGATTCCHAR:"desc",0,3,4,1,0x2901
+BLEGATTCCHAR:"char",0,3,5,0xC304,0x08
+BLEGATTCCHAR:"char",0,3,6,0xC305,0x10
+BLEGATTCCHAR:"desc",0,3,6,1,0x2902
+BLEGATTCCHAR:"char",0,3,7,0xC306,0x20
+BLEGATTCCHAR:"desc",0,3,7,1,0x2902
```

(续下页)

(接上页)

```
+BLEGATTCCCHAR:"char",0,3,8,0xC307,0x02
+BLEGATTCCCHAR:"desc",0,3,8,1,0x2901

OK
```

6. ESP32-C6 Bluetooth LE 客户端读取一个特征值。

命令：

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

响应：

```
+BLEGATTCRD:0,1,0

OK
```

说明：

- 请注意目标特征值必须要有读权限。
- 如果 ESP32-C6 Bluetooth LE 客户端读取特征成功，ESP32-C6 Bluetooth LE 服务端则会提示 +READ:0,"7c:df:a1:b3:8d:de"。

7. ESP32-C6 Bluetooth LE 客户端写一个特征值。

命令：

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

响应：

```
>
```

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

```
OK
```

说明：

- 如果 ESP32-C6 Bluetooth LE 客户端写特征描述符成功，ESP32-C6 Bluetooth LE 服务端则会提示 +WRITE:<conn_index>,<srv_index>,<char_index>,[<desc_index>],<len>,<value>。

8. Indicate 一个特征值。

ESP32-C6 Bluetooth LE 客户端：

命令：

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

响应：

```
>
```

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

为了接收 ESP32-C6 Bluetooth LE 服务端发送过来的数据（通过 notify 方式或者 indicate 方式），ESP32-C6 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。

```
OK
```

说明：

- 如果 ESP32-C6 Bluetooth LE 客户端写特征描述符成功，ESP32-C6 Bluetooth LE 服务端则会提示 +WRITE:<conn_index>,<srv_index>,<char_index>,<desc_index>,<len>,<value>。

ESP32-C6 Bluetooth LE 服务端：

命令：

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

响应：

```
>
```

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

```
OK
```

说明：

- 如果 ESP32-C6 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，ESP32-C6 Bluetooth LE 客户端的 <srv_index> 值等于 ESP32-C6 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，你可参考文档[如何自定义低功耗蓝牙服务](#)。

4.4.3 Bluetooth LE 服务端读写服务特征值

以下示例同时使用两块 ESP32-C6 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 ESP32-C6 Bluetooth LE 服务端开头的操作只需要在 ESP32-C6 Bluetooth LE 服务端执行即可，以 ESP32-C6 Bluetooth LE 客户端开头的操作只需要在 ESP32-C6 Bluetooth LE 客户端执行即可。

1. 初始化 Bluetooth LE 功能。

ESP32-C6 Bluetooth LE 服务端：

命令：

```
AT+BLEINIT=2
```

响应：

```
OK
```

ESP32-C6 Bluetooth LE 客户端：

命令：

```
AT+BLEINIT=1
```

响应：

```
OK
```

1. ESP32-C6 Bluetooth LE 服务端获取其 MAC 地址。

命令：

```
AT+BLEADDR?
```

响应：

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

说明：

- 你查询到的地址可能与上述响应中的不同，请记住你的地址，下面的步骤中会用到。

2. ESP32-C6 Bluetooth LE 服务端设置广播参数。

命令：

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

响应:

```
OK
```

3. ESP32-C6 Bluetooth LE 服务端设置广播数据。

命令:

```
AT+BLEADVDATA="0201060A09457370726573736966030302A0"
```

响应:

```
OK
```

4. ESP32-C6 Bluetooth LE 服务端开始广播。

命令:

```
AT+BLEADVSTART
```

响应:

```
OK
```

1. ESP32-C6 Bluetooth LE 客户端获取其 MAC 地址。

命令:

```
AT+BLEADDR?
```

响应:

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

```
OK
```

说明:

- 你查询到的地址可能与上述响应中的不同, 请记住你的地址, 下面的步骤中会用到。

2. ESP32-C6 Bluetooth LE 客户端开始扫描, 持续 3 秒。

命令:

```
AT+BLES SCAN=1,3
```

响应:

```
OK
```

```
+BLES SCAN:"24:0a:c4:d6:e4:46",-78,0201060a09457370726573736966030302a0,,0
```

```
+BLES SCAN:"45:03:cb:ac:aa:a0",-62,0201060aff4c001005441c61df7d,,1
```

```
+BLES SCAN:"24:0a:c4:d6:e4:46",-26,0201060a09457370726573736966030302a0,,0
```

说明:

- 你的扫描结果可能与上述响应中的不同。

3. 建立 the Bluetooth LE 连接。

ESP32-C6 Bluetooth LE 客户端:

命令:

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

响应:

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

```
OK
```

说明:

- 输入上述命令时, 请使用你的 ESP32-C6 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。

ESP32-C6 Bluetooth LE 服务端:

命令:

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

响应:

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

```
OK
```

说明:

- 输入上述命令时, 请使用你的 ESP32-C6 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. ESP32-C6 Bluetooth LE 客户端查询本地服务。

命令:

```
AT+BLEGATTSSRV?
```

响应:

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

```
OK
```

2. ESP32-C6 Bluetooth LE 客户端发现本地特征。

命令:

```
AT+BLEGATTCHAR?
```

响应:

```
+BLEGATTCHAR:"char",1,1,0xC300,0x02
+BLEGATTCHAR:"desc",1,1,1,0x2901
+BLEGATTCHAR:"char",1,2,0xC301,0x02
+BLEGATTCHAR:"desc",1,2,1,0x2901
+BLEGATTCHAR:"char",1,3,0xC302,0x08
+BLEGATTCHAR:"desc",1,3,1,0x2901
+BLEGATTCHAR:"char",1,4,0xC303,0x04
+BLEGATTCHAR:"desc",1,4,1,0x2901
+BLEGATTCHAR:"char",1,5,0xC304,0x08
+BLEGATTCHAR:"char",1,6,0xC305,0x10
+BLEGATTCHAR:"desc",1,6,1,0x2902
+BLEGATTCHAR:"char",1,7,0xC306,0x20
+BLEGATTCHAR:"desc",1,7,1,0x2902
+BLEGATTCHAR:"char",1,8,0xC307,0x02
+BLEGATTCHAR:"desc",1,8,1,0x2901
+BLEGATTCHAR:"char",2,1,0xC400,0x02
+BLEGATTCHAR:"desc",2,1,1,0x2901
+BLEGATTCHAR:"char",2,2,0xC401,0x02
+BLEGATTCHAR:"desc",2,2,1,0x2901
```

```
OK
```

3. ESP32-C6 Bluetooth LE 服务端发现对端服务。

命令:

```
AT+BLEGATTCPRIMSRV=0
```

响应:

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

OK

说明:

- ESP32-C6 Bluetooth LE 服务端查询服务的结果, 比 ESP32-C6 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, ESP32-C6 Bluetooth LE 服务端查询的 <srv_index> 值等于 ESP32-C6 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 ESP32-C6 Bluetooth LE 服务端查询到的 <srv_index> 为 3, 如果在 ESP32-C6 Bluetooth LE 服务端通过 [AT+BLEGATTSSRV?](#) 命令查询, 则 <srv_index> 为 1。

4. ESP32-C6 Bluetooth LE 服务端发现对端特征。

命令:

```
AT+BLEGATTCCCHAR=0,3
```

响应:

```
+BLEGATTCCCHAR:"char",0,3,1,0xC300,0x02
+BLEGATTCCCHAR:"desc",0,3,1,1,0x2901
+BLEGATTCCCHAR:"char",0,3,2,0xC301,0x02
+BLEGATTCCCHAR:"desc",0,3,2,1,0x2901
+BLEGATTCCCHAR:"char",0,3,3,0xC302,0x08
+BLEGATTCCCHAR:"desc",0,3,3,1,0x2901
+BLEGATTCCCHAR:"char",0,3,4,0xC303,0x04
+BLEGATTCCCHAR:"desc",0,3,4,1,0x2901
+BLEGATTCCCHAR:"char",0,3,5,0xC304,0x08
+BLEGATTCCCHAR:"char",0,3,6,0xC305,0x10
+BLEGATTCCCHAR:"desc",0,3,6,1,0x2902
+BLEGATTCCCHAR:"char",0,3,7,0xC306,0x20
+BLEGATTCCCHAR:"desc",0,3,7,1,0x2902
+BLEGATTCCCHAR:"char",0,3,8,0xC307,0x02
+BLEGATTCCCHAR:"desc",0,3,8,1,0x2901
```

OK

5. ESP32-C6 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。

命令:

```
AT+BLEGATTSSSETATTR=1,8,,1
```

响应:

>

命令:

```
写入一个字节 ``9``
```

响应:

OK

6. ESP32-C6 Bluetooth LE 服务端读服务特征值。

命令:

```
AT+BLEGATTCCRD=0,3,8,
```

响应:

```
+BLEGATTCRD:0,1,9
```

```
OK
```

7. ESP32-C6 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。

命令：

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

响应：

```
>
```

命令：

```
写入2个字节 ``12``
```

响应：

```
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. ESP32-C6 Bluetooth LE 客户端 notify 服务特征值

命令：

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

响应：

```
>
```

命令：

```
写入 ``1234567890`` 10个字节
```

响应：

```
OK
```

说明：

- 如果 ESP32-C6 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。

4.4.4 Bluetooth LE 连接加密

以下示例同时使用两块 ESP32-C6 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 ESP32-C6 Bluetooth LE 服务端开头的操作只需要在 ESP32-C6 Bluetooth LE 服务端执行即可，以 ESP32-C6 Bluetooth LE 客户端开头的操作只需要在 ESP32-C6 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。
- ESP-AT 最多允许绑定 10 个设备。

1. 初始化 Bluetooth LE 功能。

ESP32-C6 Bluetooth LE 服务端：

命令：

```
AT+BLEINIT=2
```

响应:

```
OK
```

ESP32-C6 Bluetooth LE 客户端:

命令:

```
AT+BLEINIT=1
```

响应:

```
OK
```

2. ESP32-C6 Bluetooth LE 服务端获取 Bluetooth LE 地址。

命令:

```
AT+BLEADDR?
```

响应:

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

说明:

- 你查询到的地址可能与上述响应中的不同, 请记住你的地址, 下面的步骤中会用到。

1. ESP32-C6 Bluetooth LE 服务端发现服务特征。

命令:

```
AT+BLEGATTCHAR?
```

响应:

```
+BLEGATTCHAR:"char",1,1,0xC300,0x02  
+BLEGATTCHAR:"desc",1,1,1,0x2901  
+BLEGATTCHAR:"char",1,2,0xC301,0x02  
+BLEGATTCHAR:"desc",1,2,1,0x2901  
+BLEGATTCHAR:"char",1,3,0xC302,0x08  
+BLEGATTCHAR:"desc",1,3,1,0x2901  
+BLEGATTCHAR:"char",1,4,0xC303,0x04  
+BLEGATTCHAR:"desc",1,4,1,0x2901  
+BLEGATTCHAR:"char",1,5,0xC304,0x08  
+BLEGATTCHAR:"char",1,6,0xC305,0x10  
+BLEGATTCHAR:"desc",1,6,1,0x2902  
+BLEGATTCHAR:"char",1,7,0xC306,0x20  
+BLEGATTCHAR:"desc",1,7,1,0x2902  
+BLEGATTCHAR:"char",1,8,0xC307,0x02  
+BLEGATTCHAR:"desc",1,8,1,0x2901  
+BLEGATTCHAR:"char",2,1,0xC400,0x02  
+BLEGATTCHAR:"desc",2,1,1,0x2901  
+BLEGATTCHAR:"char",2,2,0xC401,0x02  
+BLEGATTCHAR:"desc",2,2,1,0x2901  
  
OK
```

2. ESP32-C6 Bluetooth LE 服务端开始广播, 之后 ESP32-C6 Bluetooth LE 客户端开始扫描并且持续 3 秒钟。

ESP32-C6 Bluetooth LE 服务端:

命令:

```
AT+BLEADVSTART
```

响应:

OK

ESP32-C6 Bluetooth LE 客户端:

命令:

AT+BLESCAN=1,3

响应:

OK

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

说明:

- 你的扫描结果可能与上述响应中的不同。

3. 建立 Bluetooth LE 连接。

ESP32-C6 Bluetooth LE 客户端:

命令:

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

响应:

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

OK

说明:

- 输入上述命令时, 请使用你的 ESP32-C6 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。

4. ESP32-C6 Bluetooth LE 客户端发现服务。

命令:

AT+BLEGATTCPRIMSRV=0

响应:

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

OK

说明:

- ESP32-C6 Bluetooth LE 客户端查询服务的结果, 比 ESP32-C6 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, ESP32-C6 Bluetooth LE 客户端查询的 <srv_index> 值等于 ESP32-C6 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 ESP32-C6 Bluetooth LE 客户端查询到的 <srv_index> 为 3, 如果在 ESP32-C6 Bluetooth LE 服务端通过 [AT+BLEGATTSSRV?](#) 命令查询, 则 <srv_index> 为 1。

5. ESP32-C6 Bluetooth LE 客户端发现特征值。

命令:

AT+BLEGATTCCHAR=0,3

响应:

```
+BLEGATTCCHAR:"char",0,3,1,0xC300,0x02
+BLEGATTCCHAR:"desc",0,3,1,1,0x2901
+BLEGATTCCHAR:"char",0,3,2,0xC301,0x02
+BLEGATTCCHAR:"desc",0,3,2,1,0x2901
```

(续下页)

(接上页)

```
+BLEGATTCCCHAR:"char",0,3,3,0xC302,0x08
+BLEGATTCCCHAR:"desc",0,3,3,1,0x2901
+BLEGATTCCCHAR:"char",0,3,4,0xC303,0x04
+BLEGATTCCCHAR:"desc",0,3,4,1,0x2901
+BLEGATTCCCHAR:"char",0,3,5,0xC304,0x08
+BLEGATTCCCHAR:"char",0,3,6,0xC305,0x10
+BLEGATTCCCHAR:"desc",0,3,6,1,0x2902
+BLEGATTCCCHAR:"char",0,3,7,0xC306,0x20
+BLEGATTCCCHAR:"desc",0,3,7,1,0x2902
+BLEGATTCCCHAR:"char",0,3,8,0xC307,0x02
+BLEGATTCCCHAR:"desc",0,3,8,1,0x2901
```

OK

6. 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。

ESP32-C6 Bluetooth LE 服务端:

命令:

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

响应:

OK

ESP32-C6 Bluetooth LE 客户端:

命令:

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

响应:

OK

说明:

- 在本例中, ESP32-C6 Bluetooth LE 服务端输入配对码, ESP32-C6 Bluetooth LE 客户端显示配对码。
- ESP-AT 支持 Legacy Pairing 和 Secure Connections 两种加密方式, 但后者有更高优先级的优先级。如果对端也支持 Secure Connections, 则会采用 Secure Connections 方式加密。

7. ESP32-C6 Bluetooth LE 客户端发起加密请求。

命令:

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

响应:

OK

说明:

如果 ESP32-C6 Bluetooth LE 服务端成功接收到加密请求, ESP32-C6 Bluetooth LE 服务端则会提示 +BLESECREQ:0。

1. ESP32-C6 Bluetooth LE 客户端回复配对码。

命令:

```
AT+BLEKEYREPLY=0,793718
```

响应:

OK

执行这个命令之后, 在 ESP32-C6 Bluetooth LE 服务端和 ESP32-C6 Bluetooth LE 客户端会有一些对应信息提示。

ESP32-C6 Bluetooth LE 服务端:

```
+BLESECKEYTYPE:0,16
+BLESECKEYTYPE:0,1
+BLESECKEYTYPE:0,32
+BLESECKEYTYPE:0,2
+BLEAUTHCMPL:0,0
```

ESP32-C6 Bluetooth LE 客户端:

```
+BLESECNTFYKEY:0,793718
+BLESECKEYTYPE:0,2
+BLESECKEYTYPE:0,16
+BLESECKEYTYPE:0,1
+BLESECKEYTYPE:0,32
+BLEAUTHCMPL:0,0
```

你可以忽略以 +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。

4.4.5 两个 ESP32-C6 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 ESP32-C6 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 ESP32-C6 Bluetooth LE 服务端开头的操作只需要在 ESP32-C6 Bluetooth LE 服务端执行即可，以 ESP32-C6 Bluetooth LE 客户端开头的操作只需要在 ESP32-C6 Bluetooth LE 客户端执行即可。

1. 初始化 Bluetooth LE 功能。

ESP32-C6 Bluetooth LE 服务端:

命令:

```
AT+BLEINIT=2
```

响应:

```
OK
```

ESP32-C6 Bluetooth LE 客户端:

命令:

```
AT+BLEINIT=1
```

响应:

```
OK
```

1. ESP32-C6 Bluetooth LE 服务端获取其 MAC 地址。

命令:

```
AT+BLEADDR?
```

响应:

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

说明:

- 你查询到的地址可能与上述响应中的不同，请记住你的地址，下面的步骤中会用到。

2. ESP32-C6 Bluetooth LE 服务端设置广播参数。

命令：

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

响应：

```
OK
```

3. ESP32-C6 Bluetooth LE 服务端设置广播数据。

命令：

```
AT+BLEADVDATA="0201060A09457370726573736966030302A0"
```

响应：

```
OK
```

4. ESP32-C6 Bluetooth LE 服务端开始广播。

命令：

```
AT+BLEADVSTART
```

响应：

```
OK
```

5. ESP32-C6 Bluetooth LE 客户端开始扫描，持续 3 秒。

命令：

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

响应：

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

说明：

- 你的扫描结果可能与上述响应中的不同。

6. 建立 the Bluetooth LE 连接。

ESP32-C6 Bluetooth LE 客户端：

命令：

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

响应：

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

```
OK
```

说明：

- 输入上述命令时，请使用你的 ESP32-C6 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。

7. ESP32-C6 Bluetooth LE 服务端查询服务。

命令：

```
AT+BLEGATTSSRV?
```

响应：

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

```
OK
```

8. ESP32-C6 Bluetooth LE 服务端发现特征。

命令：

AT+BLEGATTCHAR?

响应：

```

+BLEGATTCHAR:"char",1,1,0xC300,0x02
+BLEGATTCHAR:"desc",1,1,1,0x2901
+BLEGATTCHAR:"char",1,2,0xC301,0x02
+BLEGATTCHAR:"desc",1,2,1,0x2901
+BLEGATTCHAR:"char",1,3,0xC302,0x08
+BLEGATTCHAR:"desc",1,3,1,0x2901
+BLEGATTCHAR:"char",1,4,0xC303,0x04
+BLEGATTCHAR:"desc",1,4,1,0x2901
+BLEGATTCHAR:"char",1,5,0xC304,0x08
+BLEGATTCHAR:"char",1,6,0xC305,0x10
+BLEGATTCHAR:"desc",1,6,1,0x2902
+BLEGATTCHAR:"char",1,7,0xC306,0x20
+BLEGATTCHAR:"desc",1,7,1,0x2902
+BLEGATTCHAR:"char",1,8,0xC307,0x02
+BLEGATTCHAR:"desc",1,8,1,0x2901
+BLEGATTCHAR:"char",2,1,0xC400,0x02
+BLEGATTCHAR:"desc",2,1,1,0x2901
+BLEGATTCHAR:"char",2,2,0xC401,0x02
+BLEGATTCHAR:"desc",2,2,1,0x2901

```

OK

9. ESP32-C6 Bluetooth LE 客户端发现服务。

命令：

AT+BLEGATTCPRIMSRV=0

响应：

```

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

```

OK

说明：

- ESP32-C6 Bluetooth LE 客户端查询服务的结果，比 ESP32-C6 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，ESP32-C6 Bluetooth LE 客户端查询的 <srv_index> 值等于 ESP32-C6 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 ESP32-C6 Bluetooth LE 客户端查询到的 <srv_index> 为 3，如果在 ESP32-C6 Bluetooth LE 服务端通过 [AT+BLEGATTSSRV?](#) 命令查询，则 <srv_index> 为 1。

10. ESP32-C6 Bluetooth LE 客户端发现特征。

命令：

AT+BLEGATTCCHAR=0,3

响应：

```

+BLEGATTCCHAR:"char",0,3,1,0xC300,0x02
+BLEGATTCCHAR:"desc",0,3,1,1,0x2901
+BLEGATTCCHAR:"char",0,3,2,0xC301,0x02
+BLEGATTCCHAR:"desc",0,3,2,1,0x2901
+BLEGATTCCHAR:"char",0,3,3,0xC302,0x08
+BLEGATTCCHAR:"desc",0,3,3,1,0x2901
+BLEGATTCCHAR:"char",0,3,4,0xC303,0x04

```

(续下页)

(接上页)

```
+BLEGATTCCCHAR:"desc",0,3,4,1,0x2901
+BLEGATTCCCHAR:"char",0,3,5,0xC304,0x08
+BLEGATTCCCHAR:"char",0,3,6,0xC305,0x10
+BLEGATTCCCHAR:"desc",0,3,6,1,0x2902
+BLEGATTCCCHAR:"char",0,3,7,0xC306,0x20
+BLEGATTCCCHAR:"desc",0,3,7,1,0x2902
+BLEGATTCCCHAR:"char",0,3,8,0xC307,0x02
+BLEGATTCCCHAR:"desc",0,3,8,1,0x2901
```

OK

11. ESP32-C6 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。

命令：

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

响应：

OK

12. ESP32-C6 Bluetooth LE 客户端使能 Bluetooth LE SPP。

命令：

```
AT+BLESPP
```

响应：

OK

>

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：

- ESP32-C6 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 ESP32-C6 Bluetooth LE 服务端。

13. ESP32-C6 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。

命令：

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

响应：

OK

14. ESP32-C6 Bluetooth LE 服务端使能 Bluetooth LE SPP。

命令：

```
AT+BLESPP
```

响应：

OK

>

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：

- ESP32-C6 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 ESP32-C6 Bluetooth LE 客户端。
- 如果 ESP32-C6 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 ESP32-C6 Bluetooth LE 客户端需要先开启侦听 Notify 或者 Indicate。

例如, ESP32-C6 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, ESP32-C6 Bluetooth LE 服务端才能成功实现透传。

- 对于同一服务, ESP32-C6 Bluetooth LE 客户端的 `<srv_index>` 值等于 ESP32-C6 Bluetooth LE 服务端的 `<srv_index>` 值 + 2, 这是正常现象。

4.4.6 ESP32-C6 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 ESP32-C6 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 ESP32-C6 Bluetooth LE 服务端开头的操作只需要在 ESP32-C6 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。

ESP32-C6 Bluetooth LE 服务端:

命令:

```
AT+BLEINIT=2
```

响应:

```
OK
```

1. ESP32-C6 Bluetooth LE 服务端获取其 MAC 地址。

命令:

```
AT+BLEADDR?
```

响应:

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

说明:

- 你查询到的地址可能与上述响应中的不同, 请记住你的地址, 下面的步骤中会用到。

2. ESP32-C6 Bluetooth LE 服务端设置广播参数。

命令:

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

响应:

```
OK
```

3. ESP32-C6 Bluetooth LE 服务端设置广播数据。

命令:

```
AT+BLEADVDATA="0201060A09457370726573736966030302A0"
```

响应:

```
OK
```

4. ESP32-C6 Bluetooth LE 服务端开始广播。

命令:

```
AT+BLEADVSTART
```

响应:

OK

5. 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 ESP32-C6 Bluetooth LE 服务端的 MAC 地址，点击 CONNECT 进行连接。此时 ESP32-C6 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。

6. ESP32-C6 Bluetooth LE 服务端查询服务。

命令：

AT+BLEGATTSSRV?

响应：

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

OK

7. ESP32-C6 Bluetooth LE 服务端发现特征。

命令：

AT+BLEGATTCHAR?

响应：

```
+BLEGATTCHAR:"char",1,1,0xC300,0x02
+BLEGATTCHAR:"desc",1,1,1,0x2901
+BLEGATTCHAR:"char",1,2,0xC301,0x02
+BLEGATTCHAR:"desc",1,2,1,0x2901
+BLEGATTCHAR:"char",1,3,0xC302,0x08
+BLEGATTCHAR:"desc",1,3,1,0x2901
+BLEGATTCHAR:"char",1,4,0xC303,0x04
+BLEGATTCHAR:"desc",1,4,1,0x2901
+BLEGATTCHAR:"char",1,5,0xC304,0x08
+BLEGATTCHAR:"char",1,6,0xC305,0x10
+BLEGATTCHAR:"desc",1,6,1,0x2902
+BLEGATTCHAR:"char",1,7,0xC306,0x20
+BLEGATTCHAR:"desc",1,7,1,0x2902
+BLEGATTCHAR:"char",1,8,0xC307,0x02
+BLEGATTCHAR:"desc",1,8,1,0x2901
+BLEGATTCHAR:"char",2,1,0xC400,0x02
+BLEGATTCHAR:"desc",2,1,1,0x2901
+BLEGATTCHAR:"char",2,2,0xC401,0x02
+BLEGATTCHAR:"desc",2,2,1,0x2901
```

OK

8. 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 ESP-AT 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。

9. ESP32-C6 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。

命令：

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

响应：

OK

10. ESP32-C6 Bluetooth LE 服务端使能 Bluetooth LE SPP。

命令：

```
AT+BLESP
```

响应：

```
OK
>
```

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。

11. 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 ESP32-C6 Bluetooth LE 服务端，此时 ESP32-C6 Bluetooth LE 服务端可以收到 test。

12. ESP32-C6 Bluetooth LE 服务端发送数据。

在 ESP32-C6 Bluetooth LE 服务端直接发送 test，此时 LightBlue 客户端可以收到 test。

4.4.7 ESP32-C6 和手机之间建立 Bluetooth LE 连接并配对

该示例展示了如何在 ESP32-C6 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 ESP32-C6 Bluetooth LE 服务端开头的操作只需要在 ESP32-C6 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。

ESP32-C6 Bluetooth LE 服务端：
命令：

```
AT+BLEINIT=2
```

响应：

```
OK
```

1. ESP32-C6 Bluetooth LE 服务端获取其 MAC 地址。

命令：

```
AT+BLEADDR?
```

响应：

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

说明：

- 你查询到的地址可能与上述响应中的不同，请记住你的地址，下面的步骤中会用到。

2. ESP32-C6 Bluetooth LE 服务端设置广播参数。

命令：

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

响应：

```
OK
```

3. ESP32-C6 Bluetooth LE 服务端设置广播数据。

命令：

```
AT+BLEADVDATA="0201060A09457370726573736966030302A0"
```

响应：


```
OK
```

4. ESP32-C6 Bluetooth LE 服务端设置加密参数。

命令：

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

响应：

```
OK
```

5. ESP32-C6 Bluetooth LE 服务端开始广播。

命令：

```
AT+BLEADVSTART
```

响应：

```
OK
```

6. 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 ESP32-C6 Bluetooth LE 服务端的 MAC 地址，点击 CONNECT 进行连接。此时 ESP32-C6 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。

7. ESP32-C6 Bluetooth LE 服务端发起加密请求。

命令：

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

响应：

```
OK
```

8. 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 ESP32-C6 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。

9. ESP32-C6 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。

命令：

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

响应：

```
OK
```

此时 ESP32-C6 Bluetooth LE 服务端会打印类似于以下日志，这表示 ESP32-C6 Bluetooth LE 服务端与手机 Bluetooth LE 客户端完成了配对。

```
+BLESECKEYTYPE:0,16  
+BLESECKEYTYPE:0,1  
+BLESECKEYTYPE:0,32  
+BLESECKEYTYPE:0,2  
+BLEAUTHCMPL:0,0
```

4.5 MQTT AT 示例

本文档主要提供在 ESP32-C6 设备上运行 *MQTT AT 命令集* 命令的详细示例。

- 基于 *TCP* 的 *MQTT* 连接（需要本地创建 *MQTT* 代理）（适用于数据量少）
- 基于 *TCP* 的 *MQTT* 连接（需要本地创建 *MQTT* 代理）（适用于数据量多）
- 基于 *TLS* 的 *MQTT* 连接（需要本地创建 *MQTT* 代理）
- 基于 *WSS* 的 *MQTT* 连接

重要：文档中所描述的例子均基于设备已连接 Wi-Fi 的前提。

4.5.1 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 ESP32-C6 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。首先你需要创建一个本地 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 ESP32-C6 MQTT 发布者开头的操作只需要在 ESP32-C6 MQTT 发布者端执行即可，以 ESP32-C6 MQTT 订阅者开头的操作只需要在 ESP32-C6 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。

ESP32-C6 MQTT 发布者：

命令：

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

响应：

```
OK
```

ESP32-C6 MQTT 订阅者：

命令：

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

响应：

```
OK
```

2. 连接 MQTT 代理。

命令：

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

响应：

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

```
OK
```

说明：

- 你输入的 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。

ESP32-C6 MQTT 订阅者：

命令：

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

响应：

```
OK
```

4. 发布 MQTT 消息 (字符串)。

ESP32-C6 MQTT 发布者:

命令:

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

响应:

```
OK
```

说明:

- 如果 ESP32-C6 MQTT 发布者成功发布消息, 以下信息将会在 ESP32-C6 MQTT 订阅者端提示。

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

5. 关闭 MQTT 连接。

命令:

```
AT+MQTTCLEAN=0
```

响应:

```
OK
```

4.5.2 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 ESP32-C6 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。首先你需要创建一个本地 MQTT 代理, 假设 MQTT 代理的 IP 地址为 192.168.3.102, 端口为 8883。

如果你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, 则你可以使用 `AT+MQTTPUBRAW` 命令。

重要: 步骤中以 ESP32-C6 MQTT 发布者开头的操作只需要在 ESP32-C6 MQTT 发布者端执行即可, 以 ESP32-C6 MQTT 订阅者开头的操作只需要在 ESP32-C6 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。

ESP32-C6 MQTT 发布者:

命令:

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

响应:

```
OK
```

ESP32-C6 MQTT 订阅者:

命令:

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

响应:

```
OK
```

2. 连接 MQTT 代理。

命令:

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

响应:

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

OK

说明:

- 你输入的 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。

ESP32-C6 MQTT 订阅者:

命令:

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

响应:

OK

4. 发布 MQTT 消息 (字符串)。

假设你想要发布消息的数据如下, 长度为 427 字节。

```
{ "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"} }
```

ESP32-C6 MQTT 发布者:

命令:

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

响应:

OK

>

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

```
+MQTTPUB:OK
```

说明:

- 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 ESP32-C6 MQTT 发布者成功发布消息, 以下信息将会在 ESP32-C6 MQTT 订阅者端提示。

```
+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"} }
```

5. 关闭 MQTT 连接。

命令:

```
AT+MQTTCLEAN=0
```

响应:

OK

4.5.3 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 ESP32-C6 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。首先你需要创建一个本地 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 ESP32-C6 MQTT 发布者开头的操作只需要在 ESP32-C6 MQTT 发布者端执行即可，以 ESP32-C6 MQTT 订阅者开头的操作只需要在 ESP32-C6 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。

命令：

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

响应：

OK

2. 查询 SNTP 时间。

命令：

```
AT+CIPSNTPTIME?
```

响应：

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

说明：

- 你的查询 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。

ESP32-C6 MQTT 发布者：

命令：

```
AT+MQTTUSERCFG=0,4,"publisher","espressif","123456789",0,0,""
```

响应：

OK

ESP32-C6 MQTT 订阅者：

命令：

```
AT+MQTTUSERCFG=0,4,"subscriber","espressif","123456789",0,0,""
```

响应：

OK

4. 设置 MQTT 连接属性。

命令：

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

响应：

```
OK
```

5. 连接 MQTT 代理。

命令：

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

响应：

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

```
OK
```

说明：

- 你输入的 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。

ESP32-C6 MQTT 订阅者：

命令：

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

响应：

```
OK
```

7. 发布 MQTT 消息（字符串）。

ESP32-C6 MQTT 发布者：

命令：

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

响应：

```
OK
```

说明：

- 如果 ESP32-C6 MQTT 发布者成功发布消息，以下信息将会在 ESP32-C6 MQTT 订阅者端提示。

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

8. 关闭 MQTT 连接。

命令：

```
AT+MQTTCLEAN=0
```

响应：

```
OK
```

4.5.4 基于 WSS 的 MQTT 连接

以下示例同时使用两块 ESP32-C6 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 `mqtts`，端口为 8081。

重要：步骤中以 ESP32-C6 MQTT 发布者开头的操作只需要在 ESP32-C6 MQTT 发布者端执行即可，以 ESP32-C6 MQTT 订阅者开头的操作只需要在 ESP32-C6 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。

命令：

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

响应:

```
OK
```

2. 查询 SNTP 时间。

命令:

```
AT+CIPSNTPTIME?
```

响应:

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

```
OK
```

说明:

- 你的查询 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。

ESP32-C6 MQTT 发布者:

命令:

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

响应:

```
OK
```

ESP32-C6 MQTT 订阅者:

命令:

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

响应:

```
OK
```

4. 连接 MQTT 代理。

命令:

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

响应:

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

```
OK
```

说明:

- 你输入的 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。

ESP32-C6 MQTT 订阅者:

命令:

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

响应:

```
OK
```

6. 发布 MQTT 消息 (字符串)。

ESP32-C6 MQTT 发布者:

命令:

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

响应:

```
OK
```

说明:

- 如果 ESP32-C6 MQTT 发布者成功发布消息，以下信息将会在 ESP32-C6 MQTT 订阅者端提示。

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

7. 关闭 MQTT 连接。

命令:

```
AT+MQTTCLEAN=0
```

响应:

```
OK
```

4.6 MQTT AT 连接云示例

本文档主要介绍你的设备如何通过 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](#)。你需要通过阅读[AWS IoT 开发指南](#)来熟悉 AWS IoT。

请按照以下步骤使用 ESP-AT 将你的 ESP32-C6 设备连接到 AWS IoT。

- 从 [AWS IoT](#) 获取证书以及 *endpoint*
- 使用 [MQTT AT](#) 命令基于双向认证连接 [AWS IoT](#)

4.6.1 从 AWS IoT 获取证书以及 endpoint

1. 登录你的 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。

确保你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)
3. 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: 强烈建议你熟悉 [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)。
-

4.6.2 使用 MQTT AT 命令基于双向认证连接 AWS IoT

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 ESP-AT 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 ESP-AT 项目以构建 AT 固件，并将固件烧录到你的 ESP32-C6 设备。欲了解更多信息，请参阅[本地编译 ESP-AT 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

如果你已有 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](#)。

使用 AT 命令连接到 AWS IoT

1. 设置 Wi-Fi 模式为 station。

命令：

```
AT+CWMODE=1
```

响应：

```
OK
```

2. 连接 AP。

命令：

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

响应：

```
OK
```

3. 设置 SNTP Server。

命令：

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

响应：

```
OK
```

4. 查询 SNTP 时间。

命令：

```
AT+CIPSNTPTIME?
```

响应：

```
+CIPSNTPTIME:<asctime style time>
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。

命令：

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

响应：

```
OK
```

```
>
```

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。

命令：

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

响应：

```
OK
```

```
>
```

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。

命令：

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

响应：

```
OK
```

```
>
```

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。

命令：

```
AT+MQTTUSERCFG=0,5,"esp32","espressif","1234567890",0,0,""
```

响应：

```
OK
```

说明：

- AT+MQTTUSERCFG 中第二参数为 5，即双向认证，不可更改。

7. 连接 AWS IoT。

命令：

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

响应：

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

说明：

- 请在 <endpoint> 参数处填入你在 AWS IoT 控制台获取的实际 endpoint 字符串。
- 无法更改端口 8883。

8. 订阅消息。

命令：

```
AT+MQTTSUB=0,"topic/esp32at",1
```

响应：

```
OK
```

9. 发布消息。

命令：

```
AT+MQTTPUB=0,"topic/esp32at","hello aws!",1,0
```

响应：

```
+MQTTSUBRECV:0,"topic/esp32at",10,hello aws!
OK
```

示例日志

正常交互日志如下：

1. ESP32-C6 端日志
2. AWS 端日志

4.7 Web Server AT 示例

本文档主要介绍 AT web server 的使用，主要涉及以下几个方面的应用：

- 使用浏览器进行 *Wi-Fi* 配网
- 使用浏览器进行 *OTA* 固件升级
- 使用微信小程序进行 *Wi-Fi* 配网
- 使用微信小程序进行 *OTA* 固件升级
- *ESP32-C6* 使用 *Captive Portal* 功能

备注：默认的 AT 固件并不支持 AT web server 的功能，请参考[Web 服务器 AT 命令](#) 启用该功能。

4.7.1 使用浏览器进行 Wi-Fi 配网

简介

通过 web server，手机或 PC 可以设置 ESP32-C6 设备的 Wi-Fi 连接信息。你可以使用手机或电脑连接到 ESP32-C6 设备的 AP，通过浏览器打开配网网页，并将 Wi-Fi 配置信息发送给 ESP32-C6 设备，然后 ESP32-C6 设备将根据该配置信息连接到指定的路由器。

```
[20:12:43:217] AT+CWMODE=1
[20:12:43:217] OK
[20:12:43:217] AT+CWJAP="MERCURY_407",""
[20:12:58:234] WIFI CONNECTED
[20:12:58:937] WIFI GOT IP
[20:12:58:937] OK
[20:12:58:937] AT+CIPSNTPCFG=1,8,"ntp1.aliyun.com"
[20:13:01:892] OK
[20:13:01:892] AT+CIPSNTPTIME?
[20:13:10:854] +CIPSNTPTIME:Wed Jan 19 20:13:10 2022
[20:13:10:854] OK
[20:13:15:716] AT+MQTTUSERCFG=0,5,"esp32","espressif","1234567890",0,0,""
[20:13:15:716] OK
[20:13:15:716] AT+MQTTCONN=0,"a108b8vm1g634q-ats.iot.us-east-2.amazonaws.com",8883,1
[20:13:23:030] +MQTTCONNECTED:0,5,"a108b8vm1g634q-ats.iot.us-east-2.amazonaws.com",8883,"",1
[20:13:31:479] OK
[20:13:31:479] AT+MQTTSUB=0,"topic/esp32at",1
[20:13:34:275] OK
[20:13:35:521] AT+MQTTPUB=0,"topic/esp32at","hello aws!",1,0
[20:13:41:959] +MQTTSUBRECV:0,"topic/esp32at",10,hello aws!
[20:13:42:422] OK
[20:13:42:422] +MQTTSUBRECV:0,"topic/esp32at",45,{
[20:13:49:335]   "message": "Hello from AWS IoT console"
[20:13:49:335] }
```

MQTT client

Info

Connected as lotconsole-1642593579651-0

Subscriptions

Subscribe to a topic

Publish to a topic

topic/esp32at

topic/esp32at

Export

Clear

Pause

Publish

Specify a topic and a message to publish with a QoS of 0.

topic/esp32at

Publish to topic

```
1 {
2   "message": "Hello from AWS IoT console"
3 }
```

topic/esp32at

January 19, 2022, 20:13:49 (UTC+0800)

Export

Hide

```
{
  "message": "Hello from AWS IoT console"
}
```

topic/esp32at

January 19, 2022, 20:13:42 (UTC+0800)

Export

Hide

We cannot display the message as JSON, and are instead displaying it as UTF-8 String.

hello aws!

流程

整个配网流程可以分为以下三步：

- 配网设备连接 *ESP32-C6* 设备
- 使用浏览器发送配网信息
- 通知配网结果

配网设备连接 ESP32-C6 设备 首先，为了让配网设备连接 ESP32-C6 设备，ESP32-C6 设备需要配置成 AP + STA 模式，并创建一个 WEB 服务器等待配网信息，对应的 AT 命令如下：

1. 清除之前的配网信息，如果不清除配网信息，可能因为依然保留之前的配置信息从而导致 WEB 服务器无法创建。

- Command

```
AT+RESTORE
```

2. 配置 ESP32-C6 设备为 Station + SoftAP 模式。

- Command

```
AT+CWMODE=3
```

3. 设置 SoftAP 的 ssid 和 password（如设置默认连接 ssid 为 *pos_softap*，无密码的 Wi-Fi）。

- Command

```
AT+CWSAP="pos_softap","",11,0,3
```

4. 使能多连接。

- Command

```
AT+CIPMUX=1
```

5. 创建 web server，端口：80，最大连接时间：25 s（默认最大为 60 s）。

- Command

```
AT+WEBSERVER=1,80,25
```

然后，使用上述命令启动 web server 后，打开配网设备的 Wi-Fi 连接功能，连接 ESP32-C6 设备的 AP：



图 1: 连接 ESP32-C6 AP

使用浏览器发送配网信息 在配网设备连接到 ESP32-C6 设备后，即可发送 HTTP 请求，配置待接入的路由器的信息：（注意，浏览器配网不支持配网设备作为待接入 AP，例如，如果使用手机连接到 ESP32-C6 的 AP，则该手机不建议作为 ESP32-C6 设备待接入的热点。）在浏览器中输入 web server 默认的 IP 地址（如果未设置 ESP32-C6 设备的 SoftAP IP 地址，默认为 192.168.4.1，你可以通过 AT+CIPAP? 命令查询当前的 SoftAP IP 地址），打开配网界面，输入拟连接的路由器的 ssid、password，点击“开始配网”即可开始配网：



图 2: 打开配网界面

用户也可以点击配网页面中 SSID 输入框右方的下拉框，查看 ESP32-C6 模块附近的 AP 列表，选择目标 AP 并输入 password 后，点击“开始配网”即可启动配网：



图 3: 浏览器获取 Wi-Fi AP 列表示意图

通知配网结果 配网成功后网页显示如下：

说明 1：配网成功后，网页将自动关闭，若想继续访问网页，请重新输入 ESP32-C6 设备的 IP 地址，重新打开网页。

同时，在串口端将收到如下信息：

```
+WEBSERVERRSP:1      // 代表启动配网
WIFI CONNECTED       // 代表正在连接
WIFI GOT IP           // 连接成功并获取到 IP
+WEBSERVERRSP:2      // 代表网页端收到配网成功结果，此时可以释放 WEB 资源
```

如果配网失败，网页将显示：

同时，在串口端将收到如下信息：

```
+WEBSERVERRSP:1      // 代表启动配网，没有后续发起连接以及获取 IP 的信息，MCU
→ 可以在收到该条消息后建立计时，若计时超时，则配网失败。
```

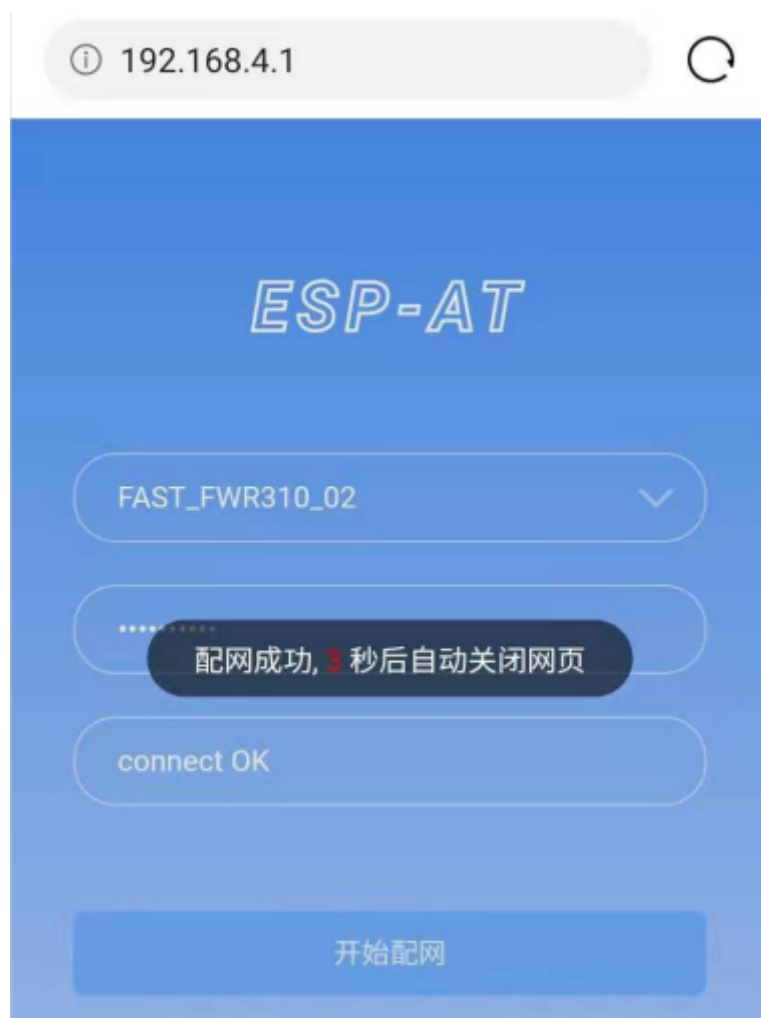



图 4: 配网成功



图 5: 配网失败

常见故障排除

说明 1: 配网页面收到提示“数据发送失败”。请检查 ESP32-C6 模块的 Wi-Fi AP 是否正确开启，以及 AP 的相关配置，并确认已经输入正确的 AT 命令成功启用 web server。

4.7.2 使用浏览器进行 OTA 固件升级

简介

浏览器打开 web server 的网页后，可以选择进入 OTA 升级页面，通过网页升级应用分区中的固件或者其它分区中的证书二进制固件（请参考文档[如何更新 PKI 配置](#)了解更多证书信息）。

流程

- 打开 OTA 配置页面
- 选择要更新的分区
- 发送新版固件
- 获取 OTA 结果

打开 OTA 配置页面 如图，点击网页右下角“OTA 升级”选项，打开 OTA 配置页面后，可以查看当前固件版本、AT Core 版本：

说明 1: 仅当浏览器连接 ESP32-C6 模块的 AP，或者访问 OTA 配置页面的设备与 ESP32-C6 模块连接在同一个子网中时，才可以打开该配置界面。

说明 2: 网页上显示的“当前固件版本”为当前用户编译的应用程序版本号，用户可通过 `./build.py menuconfig>Component config>AT>AT firmware version`（参考[本地编译 ESP-AT 工程](#)）更改该版本号，建立固件版本与应用程序的同步关系，以便于管理应用程序固件版本。

选择要更新的分区 如图，点击“Partition”下拉框来获取所有可以升级的分区：

发送新版固件 如图，点击页面中的“浏览”按钮，选择待发送的新版固件：

之后点击“固件升级”按钮发送新版固件。

说明：

- 对于 ota 分区升级，网页会对选择的固件进行检查。固件命名的后缀必须为 `.bin` 或 `.bin.xz.packed`。请确保固件的大小不要超过 `partitions_at.csv` 文件中定义的 ota 分区大小。有关此文件的详细信息，请参考文档[如何增加一个新的模组支持](#)。
- 你可以从 [GitHub Actions](#) 里下载 所需要的 OTA 固件，也可以自行[编译 ESP-AT 工程](#) 生成所需要的 OTA 固件。
- OTA 固件为 `build/esp-at.bin`（非压缩 OTA）。

备注：压缩 OTA 与非压缩 OTA 区别

- 压缩 OTA（固件后缀 `.bin.xz.packed`）：采用压缩 OTA 升级方式。芯片重启后，bootloader 会自动校验并解压固件，然后执行跳转；若校验失败，则继续使用旧版固件。解压过程发生在芯片重启后、AT 输出 `ready` 前，需要一定时间，请耐心等待。
- 非压缩 OTA（固件后缀 `.bin`）：采用非压缩 OTA 升级方式。芯片重启后，bootloader 直接校验固件并执行跳转，无需解压过程。

- 不建议升级到旧版本。降到旧版本会存在一定的兼容性问题，甚至无法运行，如果你坚持要升级到旧版本，请根据自己的产品自行测试验证功能。



图 6: OTA 配置页面



图 7: 获取所有可以升级的分区



图 8: 选择待发送的新版固件

- 对于非 ota 分区升级，网页会对选择的固件进行检查。固件命名的后缀必须为 .bin。请确保固件的大小不要超过 at_customize.csv 文件中定义的分区分大小。有关此文件的详细信息，请参考文档[如何自定义分区](#)。

获取 OTA 结果 如图，固件发送成功，将提示“升级成功”：

同时，在串口端将收到如下信息：

```
+WEBSERVERRSP:3      // 代表开始接收 OTA 固件数据
+WEBSERVERRSP:4      // 代表成功接收 OTA 固件数据
```

若接收的 OTA 固件数据失败，将提示“升级失败，请稍后重试”：

同时，在串口端将收到如下信息：

```
+WEBSERVERRSP:3      // 代表开始接收 OTA 固件数据
+WEBSERVERRSP:5      // 代表接收的 OTA 固件数据失败，用户可以选择重新打开 OTA
→ 配置界面，按照上述步骤进行 OTA 固件升级
```

说明 1：对于 ota 分区，需要执行 **AT+RST** 重启 ESP32-C6 以应用新版固件。

说明 2：ESP32-C6 会校验接收到的 ota 固件内容。但不会校验接收到的其它分区固件内容，所以请确保其它分区固件内容的正确性。

4.7.3 使用微信小程序进行 Wi-Fi 配网

简介

微信小程序配网是通过微信小程序连接 ESP32-C6 设备创建的 AP，并通过微信小程序将需要连接的 AP 信息传输给 ESP32-C6 设备，ESP32-C6 设备通过这些信息连接到对应的 AP，并通知微信小程序配网结果的解决方案。

重要：ESP-AT 微信小程序将于 2024 年 12 月 31 日下架。在此期间，请做好产品功能的过渡。如你此后仍有相关需求，敬请联系 [乐鑫商务](#)，我们将竭诚为你提供支持和解决方案，感谢你的理解和支持。

流程

整个配网流程可以分为以下四步：

- 配置 [ESP32-C6 设备参数](#)
- 加载微信小程序
- 目标 AP 选择
- 执行配网

配置 ESP32-C6 设备参数 为了让小程序连接 ESP32-C6 设备，ESP32-C6 设备需要配置成 AP + STA 模式，并创建一个 WEB 服务器等待小程序连接，对应的 AT 命令如下：

- 清除之前的配网信息，如果不清除配网信息，可能因为依然保留之前的配置信息从而导致 WEB 服务器无法创建。
 - Command

```
AT+RESTORE
```

- 配置 ESP32-C6 设备为 Station + SoftAP 模式。
 - Command



图 9: 新版固件发送成功



图 10: 新版固件发送失败

```
AT+CWMODE=3
```

3. 设置 SoftAP 的 ssid 和 password（如设置默认连接 ssid 为 *pos_softap*，password 为 *espressif*）。
 - Command

```
AT+CWSAP="pos_softap","espressif",11,3,3
```

备注：微信小程序默认向 ssid 为 *pos_softap*，password 为 *espressif* 的 SoftAP 发起连接，请确保将 ESP32-C6 设备的参数按照上述配置进行设置。

1. 使能多连接。
 - Command

```
AT+CIPMUX=1
```

2. 创建 web server，端口：80，最大连接时间：40 s（默认最大为 60 s）。
 - Command

```
AT+WEBSERVER=1,80,40
```

加载微信小程序 打开手机微信，扫描下面的二维码：



图 11: 获取小程序的二维码

打开微信小程序，进入配网界面：

目标 AP 选择 加载微信小程序后，根据待连接的目标 AP，可将配网情况分为两种情况：

1. 待接入的目标 AP 为本机配网手机提供的热点。此时请选中微信小程序页面的“本机手机热点”选项框。
2. 待接入的目标 AP 不是本机配网手机提供的热点，如路由器等 AP。此时请确保“本机手机热点”选项框未被选中。

执行配网

待接入的目标 AP 不是本机配网手机 这里以待接入的热点为路由器为例，介绍配网的过程：

1. 打开手机 Wi-Fi，连接路由器：
2. 打开微信小程序，可以看到小程序页面已经自动显示当前路由器的 ssid 为“FAST_FWR310_02”。

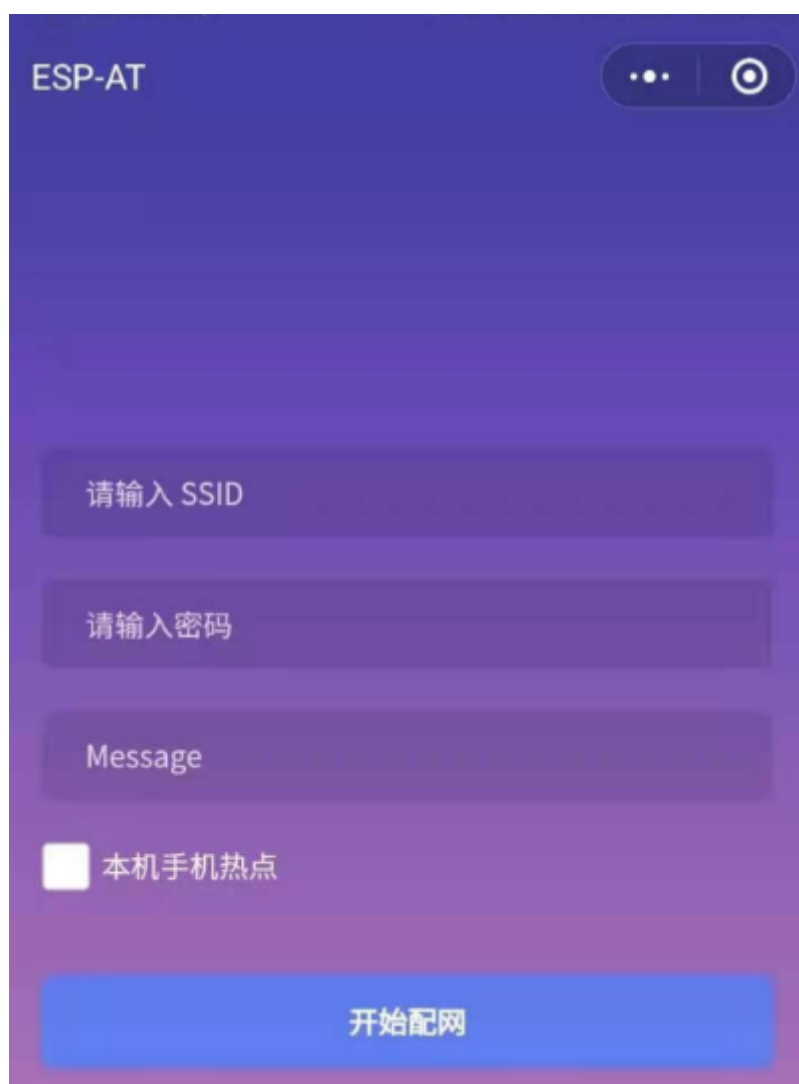


图 12: 小程序配网界面



图 13: 连接到路由器

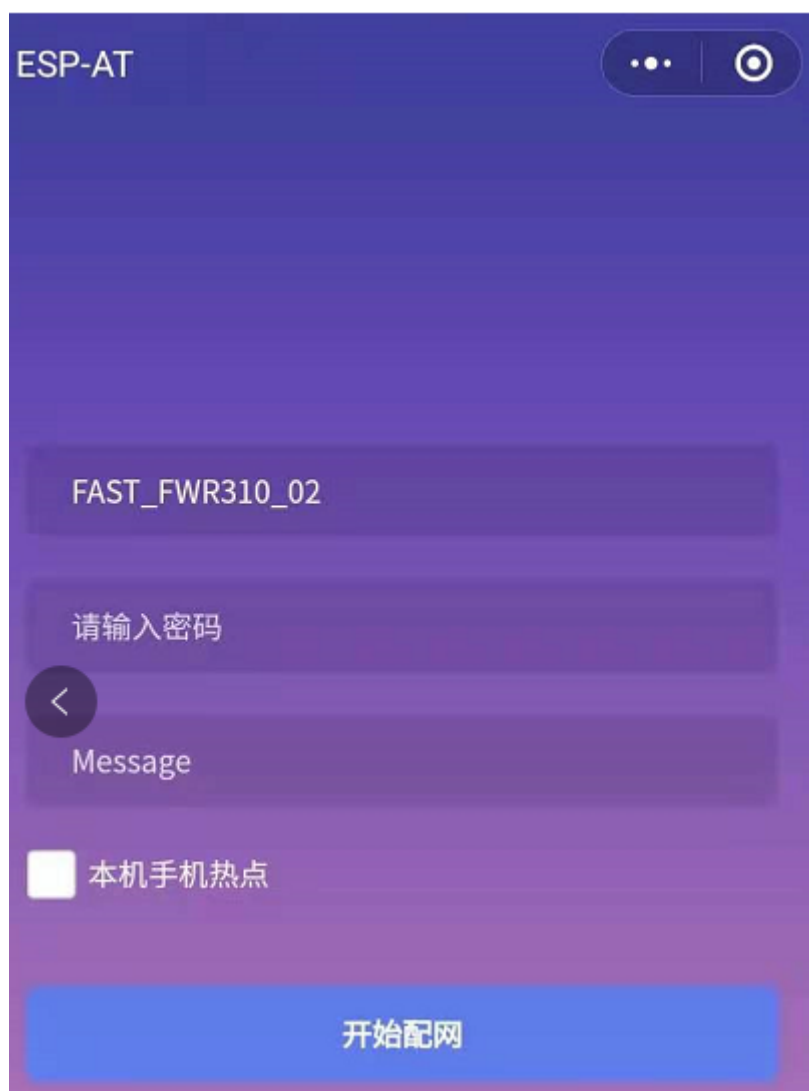


图 14: 获取路由器信息

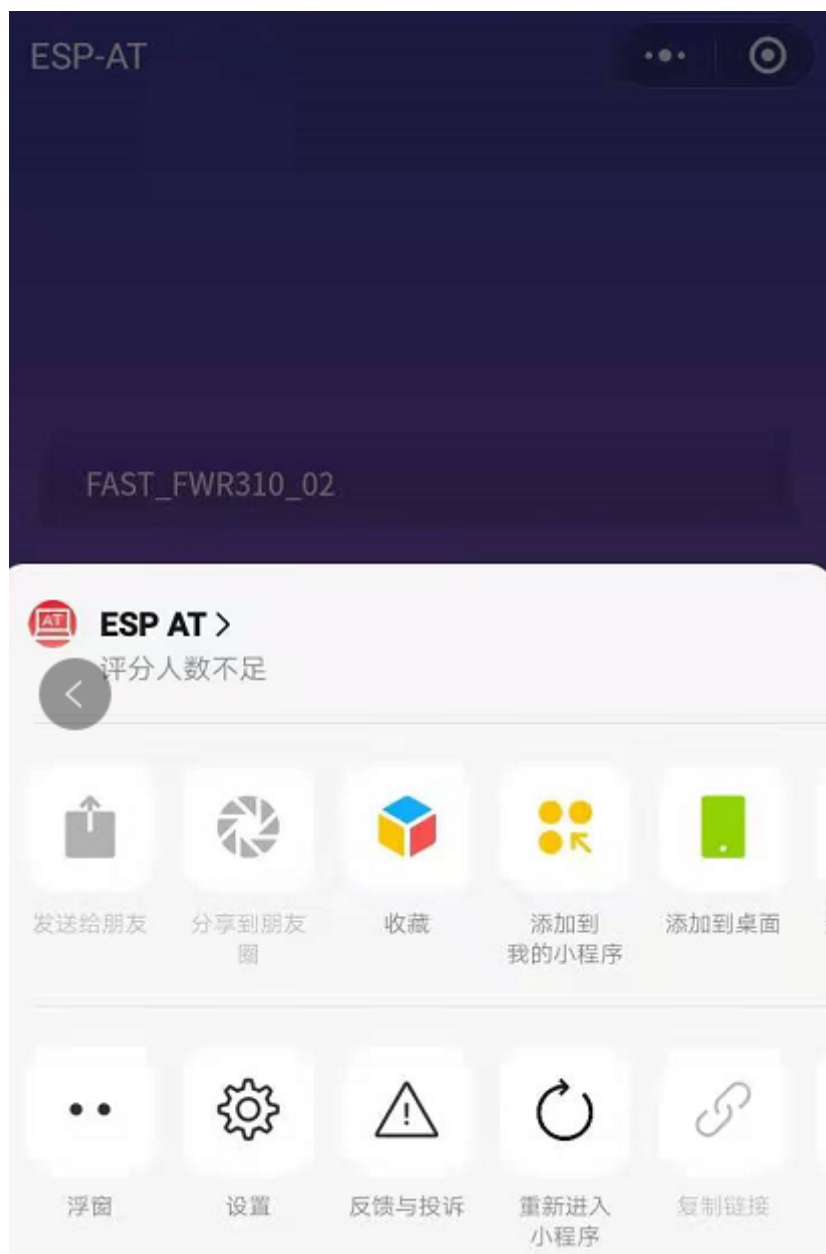


图 15: 重新进入小程序

注意：如果当前页面未显示已经连接的路由器的 ssid，请点击下图中的“重新进入小程序”，刷新当前页面：

3. 输入路由器的 password 后，点击“开始配网”。



图 16: 通过小程序连接到路由器

4. 配网成功，小程序页面显示：

同时，在串口端将收到如下信息：

```
+WEBSERVERRSP:1      // 代表启动配网
WIFI CONNECTED        // 代表正在连接
WIFI GOT IP            // 连接成功并获取到 IP
+WEBSERVERRSP:2      // 代表小程序收到配网成功结果，此时可以释放 WEB 资源
```

5. 若配网失败，则小程序页面显示：

同时，在串口端将收到如下信息：

```
+WEBSERVERRSP:1      // 代表启动配网，没有后续发起连接以及获取 IP 的信息，MCU
↪ 可以在收到该条消息后建立计时，若计时超时，则配网失败。
```

待接入的目标 AP 为本机配网手机 如果正在配网的手机作为待接入 AP，则用户不需要输入 ssid，只需要输入本机的 AP 的 password，并根据提示及时打开手机 AP 即可（如果手机支持同时打开 Wi-Fi 和分享



图 17: 通过小程序成功连接到路由器



图 18: 通过小程序连接到路由器失败

热点，也可提前打开手机 AP)。

备注：要使用该功能，手机的个人热点 MAC 地址和无线局域网 MAC 地址必须确保至少前五个字节相同。

1. 选中微信小程序页面的“本机手机热点”选项框，输入本机热点的 password 后，点击“开始配网”。

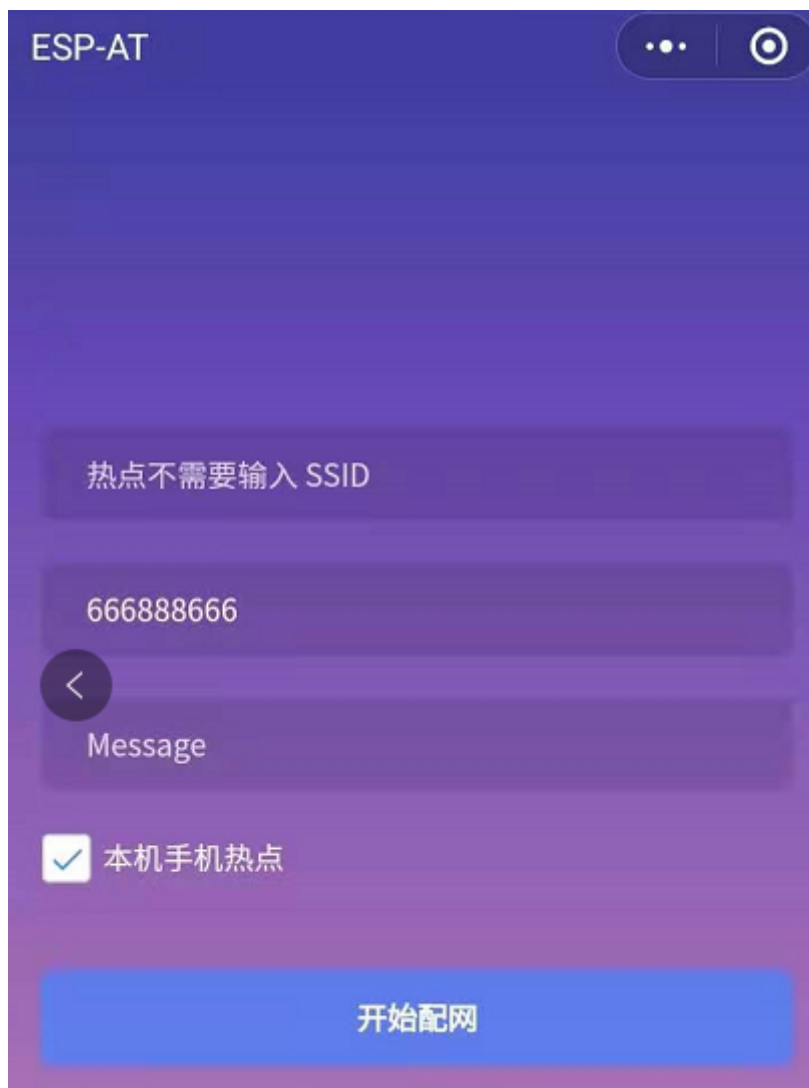


图 19: 输入 AP 的密码

2. 启动配网后，在收到提示“连接手机热点中”的提示后，请检查本机手机热点已经开启，此时 ESP32-C6 设备将自动扫描周围热点并发起连接。
3. 配网结果在小程序页面的显示以及串口端输出的数据与上述“待接入的目标 AP 不是本机配网手机”时的情况一样，请参考上文。

常见故障排除

说明 1：配网页面收到提示“数据发送失败”。请检查 ESP32-C6 模块的 Wi-Fi AP 是否正确开启，以及 AP 的相关配置，并确认已经输入正确的 AT 命令成功启用 web server。

说明 2：配网页面收到提示“连接 AP 失败”。请检查配网设备的 Wi-Fi 连接功能是否打开，检查 ESP32-C6 模块的 Wi-Fi AP 是否正确开启，以及 AP 的 ssid、password 是否按上述步骤进行配置。

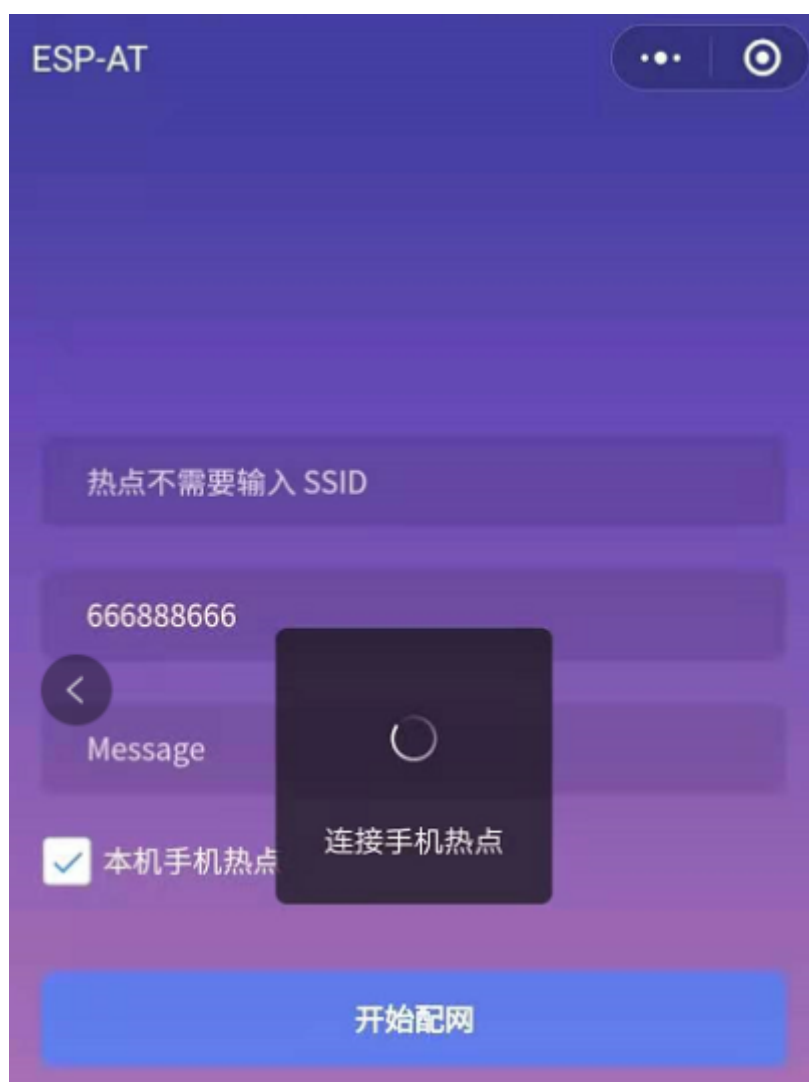


图 20: 开始连接到 AP

说明 3：配网页面收到提示“系统保存的 Wi-Fi 配置过期”。请手动使用手机连接 ESP32-C6 模块 AP，确认 ESP32-C6 模块的 ssid、password 已经按照上述步骤进行配置。

4.7.4 使用微信小程序进行 OTA 固件升级

微信小程序支持在线完成 ESP32-C6 设备的固件升级，请参考上述[配置 ESP32-C6 设备参数](#)的具体步骤完成 ESP32-C6 模块的配置（如果已经在配网时完成配置，不用重复配置）。完成配置后，设备执行 OTA 固件升级的流程与使用浏览器进行 OTA 固件升级类似，请参考[使用浏览器进行 OTA 固件升级](#)。

4.7.5 ESP32-C6 使用 Captive Portal 功能

简介

Captive Portal，是一种“强制认证主页”技术，当使用支持 Captive Portal 的 station 设备连接到提供 Captive Portal 服务的 AP 设备时，将触发 station 设备的浏览器跳转到指定的网页。更多关于 Captive Portal 的介绍，请参考 [Captive Portal Wiki](#)。

备注：默认情况下 AT web 并未启用该功能，可以通过 `./build.py menuconfig > Component config > AT > AT WEB Server command support > AT WEB captive portal support` 启用该功能，然后编译工程（请参考[本地编译 ESP-AT 工程](#)）。此外，启用该功能，可能导致使用微信小程序进行配网或 OTA 固件升级时发生页面跳转，建议仅在使用浏览器访问 AT web 时启用该功能。

流程

启用 Captive Portal 功能后，请参考上述[配网设备连接 ESP32-C6 设备](#)的具体步骤完成 ESP32-C6 模块的配置，然后连接 ESP32-C6 设备的 AP：



图 21: 连接打开 Captive Portal 功能的 AP

如上图，station 设备连接打开 Captive Portal 功能的 ESP32-C6 设备的 AP 后，提示“需登录/认证”，然后将自动打开浏览器，并跳转到 AT web 的主界面。若不能自动跳转，请根据 station 设备的提示，点击“认证”或点击上图中的“pos_softap”热点的名称，手动触发 Captive Portal 自动打开浏览器，进入到 AT web 的主界面。

常见故障排除

说明：

- 通信双方（station 设备、AP 设备）都支持 Captive Portal 功能才能保证该功能正常使用，因此，若设备连接 ESP32-C6 设备的 AP 后未提示“需登录/认证”，并且没有自动进入到 AT web 的主界面，可能是 station 设备不支持该功能，此时，请参考上述[使用浏览器发送配网信息](#)的具体步骤手动打开 AT web 的主界面。
- 某些手机（或设备）采用了激进的 Captive Portal 检测（发起大量网络连接来探测网络连通性），可能会存在兼容性问题。此时，你可以尝试增加 CONFIG_LWIP_MAX_SOCKETS 的值来解决该问题。
menuconfig 路径：Component config>LWIP>Max number of open sockets。

4.8 HTTP AT 示例

本文档主要提供在 ESP32-C6 设备上运行 [HTTP AT 命令集](#) 命令的详细示例。

- [HTTP 客户端 HEAD 请求方法](#)
- [HTTP 客户端 GET 请求方法](#)
- [HTTP 客户端 POST 请求方法（适用于 POST 少量数据）](#)
- [HTTP 客户端 POST 请求方法（推荐方式）](#)
- [HTTP 客户端 PUT 请求方法（适用于无数据情况）](#)
- [HTTP 客户端 PUT 请求方法（推荐方式）](#)
- [HTTP 客户端 DELETE 请求方法](#)

重要： 当前 ESP-AT 仅支持部分 HTTP 客户端的功能。

4.8.1 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。

命令：

```
AT+RESTORE
```

响应：

```
OK
```

2. 设置 Wi-Fi 模式为 station。

命令：

```
AT+CWMODE=1
```

响应：

```
OK
```

3. 连接路由器。

命令：

```
AT+CWJAP="espressif","1234567890"
```

响应：

```
WIFI CONNECTED
WIFI GOT IP
OK
```

说明：

- 你输入的 SSID 和密码可能跟上述命令中的不同。请使用你的路由器的 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)。

命令：

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

响应：

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

OK

说明：

- 你获取到的 HTTP 头部信息可能与上述响应中的不同。

4.8.2 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.espressif.com/sites/all/themes/espressif/images/about-us/solution-platform.jpg>。

1. 恢复出厂设置。

命令：

```
AT+RESTORE
```

响应：

OK

2. 设置 Wi-Fi 模式为 station。

命令：

```
AT+CWMODE=1
```

响应：

OK

3. 连接路由器。

命令：

```
AT+CWJAP="espressif","1234567890"
```

响应：

```
WIFI CONNECTED
WIFI GOT IP
```

OK

说明：

- 你输入的 SSID 和密码可能跟上述命令中的不同。请使用你的路由器的 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.espressif.com/sites/all/themes/espressif/images/about-us/solution-platform.jpg>, transport_type 为 2 (HTTP_TRANSPORT_OVER_SSL)。

命令：

```
AT+HTTPCLIENT=2,0,"https://www.espressif.com/sites/all/themes/espressif/images/
↳about-us/solution-platform.jpg",,,2
```

响应:

```
+HTTPCLIENT:512,<0xff><0xd8><0xff><0xe2><0x0c>XICC_PROFILE<break>
<0x01><0x01><break>
<break>
<0x0c>Hlino<0x02><0x10><break>
<break>
mnrRGB XYZ <0x07><0xce><break>
<0x02><break>
...
+HTTPCLIENT:512,<0xeb><0xe2>v<0xcb><0x98>-<0xf8><0x8a><0xae><0xf3><0xc8><0xb6>
↳<0xa8><0x86><0x02>j<0x06><0xe2>
"<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>
↳<0xcb><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><0x10><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<0x97><0xa5>P<0xdf><0xa1>#3vn<0x1b><0xc3>
↳-<0x92><0xe2>dIn<0x9c><0xb8>
<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>
aU<0xdf><0xf3><0xa3>:A<0xe2>UG<0x04>k<0xaa>*<0xa1><0xa1><0xb><0xca><0xec>
↳<0xd8>Q<0xfb><0xbc>yqY<0xec><0xfb>?<0x16>CM<0xf6>|}<0xae><0xf3><0x1e><0xdf>%
↳<0xf8><0xe8><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<0xda><0xab><0xc7>=<0xab><0xc7><0xb9>xz1f<0xba><0xfd>_e6<0xff>
↳<break>
(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>
↳<0x9e>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?<0xd7><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>1<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<0xa2><0xc2><0xb1>*<0xbc><0x13><0x13><0xa0>
↳<0xc4>)<0x83><0xb6><0xbe><0x86><0xb9><0x88>-<0x1a>
```

OK

4.8.3 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。

命令:

```
AT+RESTORE
```

响应:

OK

2. 设置 Wi-Fi 模式为 station。

命令:

```
AT+CWMODE=1
```

响应:

```
OK
```

3. 连接路由器。

命令:

```
AT+CWJAP="espressif","1234567890"
```

响应:

```
WIFI CONNECTED
WIFI GOT IP
OK
```

说明:

- 你输入的 SSID 和密码可能跟上述命令中的不同。请使用你的路由器的 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+HTTPCLIENT=3,1,"http://httpbin.org/post",,,1,"{"form":{"purpose":"test\
↪"}}"
```

响应:

```
+HTTPCLIENT:282,{
  "args": {},
  "data": "{\"form\":{\"purpose\":\"test\"}}",
  "files": {},
  "form": {},
  "headers": {
    "Content-Length": "27",
    "Content-Type": "application/json",
    "Host": "httpbin.org",
    "User-Agent": "ESP32 HTTP Client/1.0",
    "X-Amzn-Trace-Id": "Root=
+HTTPCLIENT:173,1-61503a3f-4b16b71918855b97614c5dfb"
  },
  "json": {
    "form": {
      "purpose": "test"
    }
  },
  "origin": "20.187.154.207",
  "url": "http://httpbin.org/post"
}
OK
```

说明:

- 你获取到的结果可能与上述响应中的不同。

4.8.4 HTTP 客户端 POST 请求方法 (推荐方式)

如果你 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, 则建议你使用 [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。

命令：

```
AT+RESTORE
```

响应：

```
OK
```

2. 设置 Wi-Fi 模式为 station。

命令：

```
AT+CWMODE=1
```

响应：

```
OK
```

3. 连接路由器。

命令：

```
AT+CWJAP="espressif","1234567890"
```

响应：

```
WIFI CONNECTED
WIFI GOT IP
OK
```

说明：

- 你输入的 SSID 和密码可能跟上述命令中的不同。请使用你的路由器的 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。

假设你想要 post 的 JSON 数据如下，长度为 427 字节。

```
{
  "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"
  }
}
```

命令：

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

响应：

```
OK
```

```
>
```

上述响应表示 AT 已准备好接收串行数据，此时你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。

```
+HTTPCPOST:281,{
  "args": {},
  "data": "{\n\"headers\": {\n\"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\"
  }
}"
}
```

(续下页)

(接上页)

```

→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": "ESP32 HTTP Client/1.0",
    "X-Amzn-Trace-Id": "Root=1-61505e76-278b5c267aaf55842bd58b32"
  },
  "json": {
    "headers": {
+HTTPCPOST:512,"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"
    }
  },
  "origin": "20.187.154
+HTTPCPOST:45,.207",
  "url": "http://httpbin.org/post"
}

SEND OK

```

说明:

- 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)。

4.8.5 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。

命令:

```
AT+RESTORE
```

响应:

```
OK
```

2. 设置 Wi-Fi 模式为 station。

命令:

```
AT+CWMODE=1
```

响应:

```
OK
```

3. 连接路由器。

命令:

```
AT+CWJAP="espressif","1234567890"
```

响应：

```
WIFI CONNECTED
WIFI GOT IP
OK
```

说明：

- 你输入的 SSID 和密码可能跟上述命令中的不同。请使用你的路由器的 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)。

命令：

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

响应：

```
+HTTPCLIENT:282,{
  "args": {
    "user": "foo"
  },
  "data": "",
  "files": {},
  "form": {},
  "headers": {
    "Content-Length": "0",
    "Content-Type": "application/x-www-form-urlencoded",
    "Host": "httpbin.org",
    "User-Agent": "ESP32 HTTP Client/1.0",
    "X-Amzn-Trace-Id": "R
+HTTPCLIENT:140,oat=1-61503d41-1dd8cbe0056190f721ab1912"
  },
  "json": null,
  "origin": "20.187.154.207",
  "url": "http://httpbin.org/put?user=foo"
}
OK
```

说明：

- 你获取到的结果可能与上述响应中的不同。

4.8.6 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。

命令：

```
AT+RESTORE
```

响应：

```
OK
```

2. 设置 Wi-Fi 模式为 station。

命令：

```
AT+CWMODE=1
```

响应：

```
OK
```

3. 连接路由器。

命令：

```
AT+CWJAP="espressif","1234567890"
```

响应：

```
WIFI CONNECTED
WIFI GOT IP
```

```
OK
```

说明：

- 你输入的 SSID 和密码可能跟上述命令中的不同。请使用你的路由器的 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。

假设你想要 put 的 JSON 数据如下，长度为 427 字节。

```
{ "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"} }
```

命令：

```
AT+HTTPCPUT="http://httpbin.org/put",427,2,"connection: keep-alive","content-
  ↳ type: application/json"
```

响应：

```
OK
```

```
>
```

上述响应表示 AT 已准备好接收串行数据，此时你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。

```
+HTTPCPUT:281,{
  "args": {},
  "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://httpbin.org\", \"Referer\": \"htt
+HTTPCPUT: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": "ESP32 HTTP Client/1.0",
    "X-Amzn-Trace-Id": "Root=1-635f7009-681be2d5478504dc5b83624a"
  },
  "json": {
    "headers": {
+HTTPCPUT:512,"Accept": "application/json",
```

(续下页)

(接上页)

```

"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"
    }
  },
  "origin": "52.246.135
+HTTPCPUT:43,.57",
  "url": "http://httpbin.org/put"
}

SEND OK

```

说明:

- 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)。

4.8.7 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。

- 恢复出厂设置。

命令:

```
AT+RESTORE
```

响应:

```
OK
```

- 设置 Wi-Fi 模式为 station。

命令:

```
AT+CWMODE=1
```

响应:

```
OK
```

- 连接路由器。

命令:

```
AT+CWJAP="espressif","1234567890"
```

响应:

```

WIFI CONNECTED
WIFI GOT IP
OK

```

说明:

- 你输入的 SSID 和密码可能跟上述命令中的不同。请使用你的路由器的 SSID 和密码。
- 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)。
- 命令:

```
AT+HTTPCLIENT=5,0,"https://httpbin.org/delete",,,1
```

响应：

```
+HTTPCLIENT:282,{
  "args": {},
  "data": "",
  "files": {},
  "form": {},
  "headers": {
    "Content-Length": "0",
    "Content-Type": "application/x-www-form-urlencoded",
    "Host": "httpbin.org",
    "User-Agent": "ESP32 HTTP Client/1.0",
    "X-Amzn-Trace-Id": "Root=1-61504289-468a41
+HTTPCLIENT:114,737b0d251672acec9d"
  },
  "json": null,
  "origin": "20.187.154.207",
  "url": "https://httpbin.org/delete"
}

OK
```

说明：

- 你获取到的结果可能与上述响应中的不同。

4.9 WebSocket AT 示例

本文档介绍了在 ESP32-C6 设备上运行 *WebSocket AT 命令集* 命令的详细示例。

- 基于 *TCP* 的 *WebSocket* 连接
- 基于 *TLS* 的 *WebSocket* 连接（相互鉴权）

重要： 默认的 AT 固件不支持 WebSocket 功能。如果你需要 ESP32-C6 支持 WebSocket 功能，请参考[介绍](#)。

4.9.1 基于 TCP 的 WebSocket 连接

1. 设置 Wi-Fi 模式为 station。

命令：

```
AT+CWMODE=1
```

响应：

```
OK
```

2. 连接到路由器。

命令：

```
AT+CWJAP="espressif","1234567890"
```

响应：

```
WIFI CONNECTED
WIFI GOT IP

OK
```

说明：

- 你输入的 SSID 和密码可能跟上述命令中的不同。请使用你的路由器的 SSID 和密码。
3. PC 与 ESP32-C6 设备连接同一个路由。
 4. 在 PC 上使用 python 创建一个基于 TCP 的 WebSocket 服务器。示例代码如下：

```
#!/usr/bin/env python

import asyncio
from websockets.server import serve

host = "192.168.200.249"
port = 8765

async def echo(websocket):
    async for message in websocket:
        await websocket.send(message)

async def main():
    async with serve(echo, host, port):
        print(f"Server started at ws://{host}:{port}")
        await asyncio.get_running_loop().create_future() # run forever

asyncio.run(main())
```

请修改上述代码中的 host 为 PC 的 IP 地址并保存为 ws-server.py 文件，同时运行该程序。

```
python ws-server.py
```

说明：

- 如果你的 PC 上没有安装 websockets 库，请使用 `pip install websockets` 安装。
 - 默认设置的 WebSocket 服务器端口为 8765，你也可以设置为其它端口。
5. ESP32-C6 设备作为客户端连接到 WebSocket 服务器。
- 命令：

```
AT+WSOPEN=0,"ws://192.168.200.249:8765"
```

响应：

```
+WS_CONNECTED:0

OK
```

6. 发送 4 字节数据。
- 命令：

```
AT+WSEND=0,4
```

响应：

```
OK

>
```

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。

```
SEND OK
```

说明：

- 若输入的字节数目超过 AT+WSEND 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。

由于上面的 WebSocket 服务器是一个回显服务器，因此在数据发送后，服务器也会原样返回该数据，即 AT 会输出：

```
+WS_DATA:0,4,test
```

4.9.2 基于 TLS 的 WebSocket 连接（相互鉴权）

1. 设置 Wi-Fi 模式为 station。

命令：

```
AT+CWMODE=1
```

响应：

```
OK
```

2. 连接到路由器。

命令：

```
AT+CWJAP="espressif","1234567890"
```

响应：

```
WIFI CONNECTED
WIFI GOT IP
OK
```

说明：

- 你输入的 SSID 和密码可能跟上述命令中的不同。请使用你的路由器的 SSID 和密码。

3. 设置 SNTP 服务器。

命令：

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

响应：

```
OK
```

说明：

- 你可以根据自己国家的时区设置 SNTP 服务器。

4. 查询 SNTP 时间。

命令：

```
AT+CIPSNTPTIME?
```

响应：

```
+CIPSNTPTIME:Wed Nov 13 17:00:15 2024
OK
```

说明：

- 你可以查询 SNTP 时间与实时时间是否相符来判断你设置的 SNTP 服务器是否生效。
- 设置时间是为了在 TLS 认证时校证书的有效期。

5. PC 与 ESP32-C6 设备连接同一个路由。

6. 获取 WebSocket 服务器端的 CA、证书、私钥。

你可以使用 openssl 工具生成 CA、证书、和私钥。如果遇到困难，请使用以下配置进行测试：
wss_ca.crt

```
-----BEGIN CERTIFICATE-----
MIIDNjCCA6gAwIBAgIUDe6T+Pu0BqmmTjw3s4snVRNFicMwDQYJKoZIhvcNAQEL
BQAwNjELMAkGA1UEBhMCQ04xFTATBgNVBAoMDEVUTUFJFU1NJRiBBVDEQMA4GA1UE
AwWHUm9vdCBDQTAeFw0yNDA5MTkwOTMzNDBaFw0zNDA5MTcwOTMzNDBaMDYxCzAJ
```

(续下页)

(接上页)

```
BgNVBAYTAkNOMRUwEwYDVQKDAxFU1BSRVNTSUYgQVQxEDAObgNVBAMMB1Jvb3Qg
Q0EwgGElMA0GCSqGSIB3DQEBAQUAA4IBDwAwggEKAoIBAQCylQstJGNFHLf7F+gG
oMSZSrWV4+/5Qxw1Cw3y5N1OVkMSxppYHVxj6MbOwWoCqQcx/WMTqnhg9ATDiZOE
bXFVB0aZd6EEBz24k7UvQ1ilfIw5tzmj18SSbe7CiFq302WVokBVFhSeY2jrG+sI
6PWg0WsvxzierDL9hef708IJERLX0ScBiSLZnVU4UBPTrbG2bgl3L6T5iQ95tSEX
LsmfZ41+/Q1xSC7VGH1K1WGBnUzGpv9vLc9uFGZVcAEKx1V9Y7DsyJvTosfOOMmb
yUIcBk5nVCHrfhmrfrAWWD9YaNc0gMPVp06cHzGd/Fgw6v06QshMYUOE6wxpVjb
8JYlAgMBAAGjPDA6MAwGA1UdEwQFMAMBAf8wCwYDVR0PBAQDAgGmMB0GA1UdDgQW
BBRdPJ7Nq+WXSiLN4ZLWJlQQfjrm7zANBgkqhkiG9w0BAQsFAAOCAQEANYpErE2L
IpISbzJTvG6A67YmYMyadWSNGQ2VjdLK2s3zkggF96bIZziygOa9mgAKD/yTw10t
V0xFlGUDz43DtJZ1wxo8FPrxH41cP63vjtp28+ZNkylv9Hrx8De2JjiYwRpZmlQY
EHwDEFpK26LdwPPalwdMZSyMzCtRNJ+o8Bk5kSc3V9QJmh91Le4PdfCJcfzWpSkC
dNgUMECRa4k3VFOYVUumyFofG8/kInCrogfPzZuUE20j1wChqSpOaP2CJN9QTK0q
Fn0Itrjikv6z1aTp+SJPozRPbympL9KhhhhcYJQdPaXHWYfdhEU/abnnJQpd+1Y
HPnBCb4tK/pS9w==
-----END CERTIFICATE-----
```

wss_server.crt

```
-----BEGIN CERTIFICATE-----
MIIDVTCCAj2gAwIBAgIUPLhviJh1UJDQ5Md6tHibK11jP24wDQYJKoZIhvcNAQEL
BQAwNjELMAkGA1UEBhMCQ04xFTATBgNVBAoMDEVtUfJF1U1NJRiBBVDEQMA4GA1UE
AwWHUm9vdCBDQTAeFw0yNDA5MTkwOTMzNDBaFw0zNDA5MTcwOTMzNDBaMDQxCzAJ
BgNVBAYTAkNOMRUwEwYDVQKDAxFU1BSRVNTSUYgQVQxZjAMBgNVBAMMBUwxIENB
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAtKALI+zRbUaMjukAi2ai
dzPdNdy+Tuv2K+GyB15yomIr0e1c1pfztk68zdHKcFFt0RQfz1GYtxrZzqv07o4t
K9ijfAw+k498SCTmTqHwLUKc7B9mK6UZfzSkXUnufXKE5+N96u+49e3wbk7yoNfQ
kzdtwwXUtM6ee57HvjwdKQcJXYWv/c/zVLWmAug3AEaUknS8r4LdG/X+L/bxN+9
zycL5tGgZg22KENW4QwsSUY6ntuKQDlohiNxi+wXyM3mVNOC94umICj7a10hPst
UmuLYD/gUCnM4JRvQYAVmPQC88K0Lj7aIwJedQ7TJwhJGDS6WDVEMeRfJvXswUk
4QIDAQABo10wWzAMBgNVHRMEBTADAQH/MASGA1UdDwQEAWIBpjAdBgNVHQ4EFgQU
e1wZbnvJnZr8js60EyIWiP6hRFawHwYDVR0jBBgwFoAUNrQtNxD0tapUczc8Mwln
hPZE84YwDQYJKoZIhvcNAQELBQADggEBAICnosdYBc6enaKB77AWXFzMWyDDLvd5
JSJa1IRJ+Rhs1gBxjIH66/+6QyZcJu5C/RZ+RZ04Mky3nMndc3kSCFqauBCK9xoG
/fZ5wrAfH54o4P+3Z1iGoK8EltKu1HuQYJTW8JKbLWCRJ8PnGkrDZBgSXgn8tyL4
U3hu1MdmkH//fiV0z6itfyJcZDu38718bGBhrAD3Z7gWcdIJbXxmu6m7FJadUe/N
0vsLtgkOpOyUa2rThOnJpSyXB5QnKifRGYjuPrnOIWtmADYQRUEd2zdgqRUUuYJR
ee4Vz2BFXhpZdGeD3bVAop+/YebTa0iDxXSLWkPLQfCyIkdtPXmKQPQ=
-----END CERTIFICATE-----
```

wss_server.key

```
-----BEGIN PRIVATE KEY-----
MIIEvAIBADANBgkqhkiG9w0BAQEFAASCBKYwggSiAgEAAoIBAQC0oAsj7NftRoyO
6QCLZqJ3M9013L5NS/Yr4bIHxNKiYivR7VzWl/O2TrzN0cpwUW3RFB/PUZi3GtnO
q87uji0r2KMUD6Tj3xIJ0ZOofCVQpzsH2YrpRl9lKRdSe59coTn433q77j17fBu
TvKg19CTN23DBdS0zp57nDse+PB0pBwldha/9z/NUtaYBQbcARpSSdLyvgt0b9f4
v9vE373PJwsvm0aBmDbYoQ1bhBaxJRjqe24pAOWiGI3GL7BfIzeZU05z3i6YgKPt
rU6E+y1Sa4tgP+BQKczglG9BgBwY9AKLzwqguPtojAl51DtMnCEkYNLpYNUQx5F8
lVezBStHAgMBAAECggEASkRF4FsWjyJDV91Y4nhsU6vpCCT/wCN8D/XoL9x3MOpB
jzrUAc4PoIWGXvAkFwN8LkviemlX6+2n4bDF0FN4Ij+caflQ33ZPSRCW+3zdQVnW
0MvmSorDTN3JqSvlWgI0wG3Kz8cKW2AejBR88YJbGbTgNiBXIZKVgkWC/maULKa
L2Omd5ZzF9qkWw0Bk0GxrlME+V9726L6SJHIVESkPcbojXZPM06zmz84+Zzf0UdG
pl+SocmQzo8yzhjXR8UHxtzfZbiijZGCsWiXvzqM40TaRfIBPhmmSNwp07V3lD
t6XISALqMbIHv/co5LWS5WnyQBud28MV7RpCD/tawKBgQDnGuFSmilpMR6KqRrd
O8amarsek1ND9NiWsUcXLYYHQkNC1Ttr5Gqtm3EbJbR0wLwUzL7PwgeuNmN6e6vz
IreS7wd+u03HeVVsFPeOpTlXzALMG6wuYN/Ipfe/T8v/vq9CkoZ6VdADn6OdEBAq
g6pAUQFL+uamLfrKcCiZtiOR/wKBgQDIFRhc5MyLaFlw3XumgAsh37tmlGOQ/T+K
GAWXiRhZxW340EVEzNUUXeJcGig4BgNVwXBvjYTk0unyIw72bUNrIeOqp4ZgO2vW
NUGyDgIGKZuIIGI51FeezG6Qu2s93qZTYtdjH0M4ISKgrKovY51fuXBNFSuC7BOq
i5MQamuJHwKBgHGJhiszq6aPSCbtH1KTHGwDwXwqfRfEwWd/HqLnbZJBxpPmhvPh
mvtBg5bHt1kpmv1I/XFKLMXM2KCDA54Gb1OTdQYvyjmWhX386wY8a+iTRMiLy9JZ
```

(续下页)

(接上页)

```
K3gC+a0Wge1Z+/Zj0AdnOgTLH+14y8hnOQwx/8YZNJltu2kbIwcpMV53AoGAFMzE
meepL/DoI2iS+ysifSIChFbexurc2SFIK4mwBgY3cX9NRt6qZBSifIqnlbNiU17p
r18a6qLWeTqVyp5vPMroHQyPVp+2xS0C1VlJcpSOu6cKLxLZDQQZlmg1bNghmFeV
JpPQbBxdujBYT9peON5RQ2IpBNI/9SHPZwx5I2cCgYBdCmby0loBi5lCbCXKV5c3
mz1PpM6zR3aLYsFyknWRtLv3UPMGzOLsRDkwfQEMZWt/VFjXehajyqd8X7r3V0lX
cz5LwmHOMSKDYgdzrJjvE3lPxDdKtmyPr5nnF/8/SF6Lawj7v6eGnpiZNqHC/wkK
9EmCwXWJc/9GVWSZtEOz/Q==
-----END PRIVATE KEY-----
```

7. 在 PC 上使用 python 创建一个基于 TLS 的 WebSocket 服务器。示例代码如下：

```
#!/usr/bin/env python

import asyncio
import ssl
import websockets

host = '192.168.200.249'
port = 8766
wss_ca = '/your_path/wss_ca.crt'
wss_cert = '/your_path/wss_server.crt'
wss_key = '/your_path/wss_server.key'

async def echo(websocket, path):
    async for message in websocket:
        await websocket.send(message)

ssl_context = ssl.SSLContext(ssl.PROTOCOL_TLS_SERVER)
ssl_context.load_cert_chain(certfile=wss_cert, keyfile=wss_key)
ssl_context.verify_mode = ssl.CERT_REQUIRED
ssl_context.load_verify_locations(wss_ca)

start_server = websockets.serve(echo, host, port, ssl=ssl_context)

asyncio.get_event_loop().run_until_complete(start_server)
print(f"Server started at wss://{host}:{port}")

asyncio.get_event_loop().run_forever()
```

请修改上述代码中的 host 为 PC 的 IP 地址，wss_ca、wss_cert、wss_key 为服务器端的 CA、证书、和私钥路径，并保存为 wss-server.py 文件，同时运行该程序。

```
python wss-server.py
```

说明：

- 如果你的 PC 上没有安装 websockets 库，请使用 `pip install websockets` 安装。
- 设置的 WebSocket 服务器端口为 8766，你也可以设置为其它端口。

8. 配置 WebSocket 连接为相互鉴权。

命令：

```
AT+WSCFG=0,30,60,4096,3,0,0
```

响应：

```
OK
```

说明：

- 如果你想使用自己的客户端证书或者使用多套证书，有两种方式：
 - 运行时更新：可以直接使用 `AT+SYSMFG` 命令更新证书，无需重新编译固件。具体操作请参考 `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 wss_cert、wss_key、wss_ca 即可。
 - 预烧录证书：如果想要在编译时预烧录证书，请参考 [如何更新 PKI 配置](#)。

9. ESP32-C6 设备作为客户端连接到 WebSocket 服务器。

命令：

```
AT+WSOPEN=0,"wss://192.168.200.249:8766"
```

响应：

```
+WS_CONNECTED:0
```

```
OK
```

10. 发送 4 字节数据。

命令：

```
AT+WSEND=0,4
```

响应：

```
OK
```

```
>
```

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。

```
SEND OK
```

说明：

- 若输入的字节数目超过 AT+WSEND 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。

由于上面的 WebSocket 服务器是一个回显服务器，因此在数据发送后，服务器也会原样返回该数据，即 AT 会输出：

```
+WS_DATA:0,4,test
```

4.10 Sleep AT 示例

本文档简要介绍并举例说明如何在 ESP32-C6 系列产品上使用 AT 命令设置睡眠模式。

- [简介](#)
- [在 Wi-Fi 模式下设置为 Modem-sleep 模式](#)
- [在 Wi-Fi 模式下设置为 Light-sleep 模式](#)
- [在蓝牙广播态下设置为 Light-sleep 模式](#)
- [在蓝牙连接态下设置为 Light-sleep 模式](#)
- [设置为 Deep-sleep 模式](#)

4.10.1 简介

ESP32-C6 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，ESP-AT 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，ESP32-C6 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。

ESP32-C6 的功耗，请参考 [ESP32-C6 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 ESP32-C6 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 μA 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。

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乐鑫模组进行测试。

硬件连接可参考下图。（注意，图中开发板只保留了 ESP32-C6，外围元器件均已移除。）

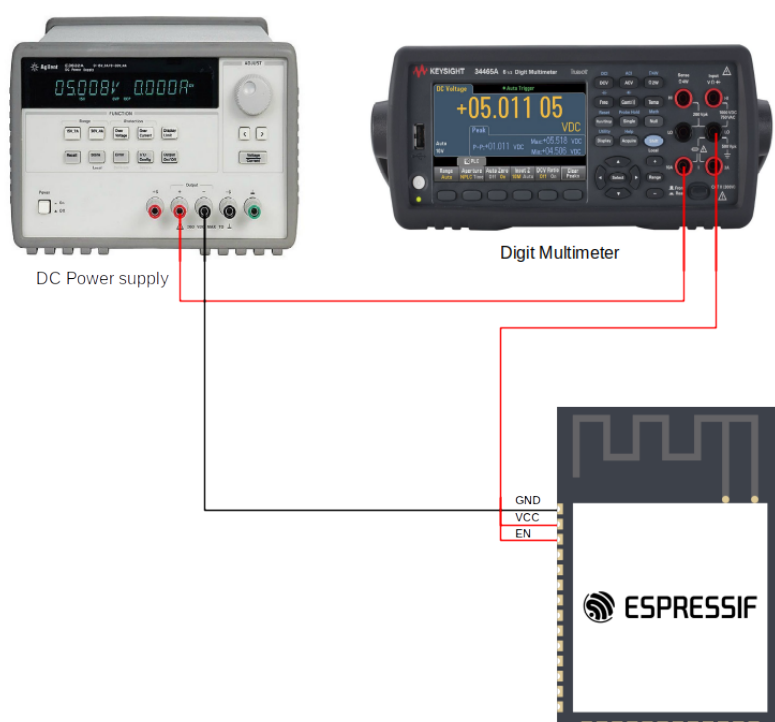


图 22: ESP32-C6 硬件连接

4.10.2 在 Wi-Fi 模式下设置为 Modem-sleep 模式

1. 设置 Wi-Fi 为 station 模式。

命令:

```
AT+CWMODE=1
```

响应:

```
OK
```

2. 连接路由器。

命令:

```
AT+CWJAP="espressif","1234567890"
```

响应:

```
WIFI CONNECTED
WIFI GOT IP

OK
```

说明:

- 你输入的 SSID 和密码可能跟上述命令中的不同。请使用你的路由器的 SSID 和密码。

3. 设置休眠模式为 Modem-sleep 模式。

命令:

```
AT+SLEEP=1
```

响应:

```
OK
```

备注: RF 将根据 AP 的 DTIM 定期关闭 (路由器一般设置 DTIM 为 1)。

4.10.3 在 Wi-Fi 模式下设置为 Light-sleep 模式

1. 设置 Wi-Fi 为 station 模式。

命令:

```
AT+CWMODE=1
```

响应:

```
OK
```

2. 连接路由器。设置监听间隔为 3。

命令:

```
AT+CWJAP="espressif","1234567890",,,3
```

响应:

```
WIFI CONNECTED
WIFI GOT IP

OK
```

说明:

- 你输入的 SSID 和密码可能跟上述命令中的不同。请使用你的路由器的 SSID 和密码。

3. 设置休眠模式为 Light-sleep 模式。

命令:

```
AT+SLEEP=2
```

响应:

```
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。

4.10.4 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。

命令：

```
AT+BLEINIT=2
```

响应：

```
OK
```

2. 设置蓝牙广播参数。设置蓝牙广播间隔为 1 s。

命令：

```
AT+BLEADVPARAM=1600,1600,0,0,7,0,0,"00:00:00:00:00:00"
```

响应：

```
OK
```

3. 开始广播。

命令：

```
AT+BLEADVSTART
```

响应：

```
OK
```

4. 禁用 Wi-Fi。

命令：

```
AT+CWINIT=0
```

响应：

```
OK
```

5. 设置休眠模式为 Light-sleep 模式。

命令：

```
AT+SLEEP=2
```

响应：

```
OK
```

4.10.5 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。

命令：

```
AT+BLEINIT=2
```

响应：

```
OK
```

2. 开始广播。

命令：

```
AT+BLEADVSTART
```

响应：

```
OK
```

3. 等待连接。

如果连接建立成功，则 AT 将会提示：

```
+BLECONN:0,"47:3f:86:dc:e4:7d"  
+BLECONNPARAM:0,0,0,6,0,500  
+BLECONNPARAM:0,0,0,24,0,500
```

```
OK
```

说明：

- 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。

4. 更新蓝牙连接参数。设置蓝牙连接间隔为 1 s。

命令：

```
AT+BLECONNPARAM=0,800,800,0,500
```

响应：

```
OK
```

如果连接参数更新成功，则 AT 将会提示：

```
+BLECONNPARAM:0,800,800,800,0,500
```

说明：

- 对于提示信息（+BLECONNPARAM），请参考[AT+BLECONNPARAM](#)获取更多信息。

5. 禁用 Wi-Fi。

命令：

```
AT+CWINIT=0
```

响应：

```
OK
```

6. 设置休眠模式为 Light-sleep 模式。

命令：

```
AT+SLEEP=2
```

响应：

OK

4.10.6 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。

命令：

```
AT+GSLP=3600000
```

响应：

OK

说明：

- 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。

4.11 AT 配网示例

本文档主要介绍以下几种 ESP-AT 支持的配网方式：

- [BluFi 配网](#)
- [SmartConfig 配网](#)
- [SoftAP 配网](#)（[WEB 配网](#)）
- [WPS 配网](#)

表 1: 配网方式关键参数总结

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。

4.11.1 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. ESP32-C6 设备设置 BluFi 名称。

命令：

```
AT+BLUFINAME="blufi_test"
```

响应：

```
OK
```

3. 开启 BluFi。

命令：

```
AT+BLUFI=1
```

响应：

```
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，ESP 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

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，ESP 设备将输出以下日志。至此，ESP 设备的配网已成功完成。

```
WIFI CONNECTED
WIFI GOT IP
```

5. （可选步骤）ESP32-C6 设备发送 BluFi 用户自定义数据。

命令：

```
AT+BLUFISEND=4
```

响应：

```
>
```

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。

```
OK
```

此时手机应用程序端显示了类似以下信息：


```
Receive custom data:
test
```

6. ESP32-C6 设备关闭 BluFi。

命令：

```
AT+BLUFI=0
```

响应：

```
+BLUFIDISCONN
```

```
OK
```

此时手机应用程序端显示了类似以下信息：

```
Disconnect <mac>, status=19
```

4.11.2 SmartConfig 配网

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。

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 ESP 设备为 Station 模式。

命令：

```
AT+CWMODE=1
```

响应：

```
OK
```

3. ESP32-C6 设备开启 SmartConfig。

命令：

```
AT+CWSTARTSMART=4,0,"1234567890123456"
```

响应：

```
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 ESP 设备端确认配网成功。

此时 ESP 设备端会输出类似如下信息：

```
smartconfig type:ESPTOUCH_V2
Smart get wifi info
ssid:test
password:1234567890
```

(续下页)

(接上页)

```
WIFI CONNECTED
WIFI GOT IP
smartconfig connected wifi
```

到此，ESP 设备的配网已成功完成。

6. ESP32-C6 设备停止 SmartConfig。

命令：

```
AT+CWSTOPSMART
```

响应：

```
OK
```

AirKiss 配网示例

1. 打开手机微信，搜索并关注“乐鑫信息科技”公众号。

2. 设置 ESP 设备为 Station 模式。

命令：

```
AT+CWMODE=1
```

响应：

```
OK
```

3. ESP32-C6 设备开启 SmartConfig。

命令：

```
AT+CWSTARTSMART=2
```

响应：

```
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“乐鑫信息科技”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 ESP 设备端会输出类似如下信息：

```
smartconfig type:AIRKISS
Smart get wifi info
ssid:test
password:1234567890
WIFI CONNECTED
WIFI GOT IP
smartconfig connected wifi
```

到此，ESP 设备的配网已成功完成。

5. ESP32-C6 设备停止 SmartConfig。

命令：

```
AT+CWSTOPSMART
```

响应：

```
OK
```

4.11.3 SoftAP 配网

AT SoftAP 配网指的是 WEB 配网。有关详细信息，请参见 [Web Server AT 示例](#)。

4.11.4 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 ESP 设备为 Station 模式。

命令：

```
AT+CWMODE=1
```

响应：

```
OK
```

3. ESP32-C6 设备开启 WPS 配网。

命令：

```
AT+WPS=1
```

响应：

```
OK
```

此时根据路由器的使用手册，开启 WPS 功能，稍等片刻，ESP 设备端会输出类似如下信息，即设备配网已成功。

```
WIFI CONNECTED
WIFI GOT IP
```

4. 查询 ESP32-C6 设备连接上的 Wi-Fi 信息。

命令：

```
AT+CWJAP?
```

响应：

```
+CWJAP:"test",.....
OK
```

5. ESP32-C6 设备关闭 WPS 配网。

命令：

```
AT+WPS=0
```

响应：

```
OK
```

Chapter 5

ESP-AT 版本简介

本文档主要介绍了 ESP-AT 的版本、如何选择版本、版本管理、支持期限、查看版本、和订阅 AT 发布等内容。

5.1 发布版本

ESP-AT 在 GitHub 平台上的完整发布历史请见 [发布说明页面](#)。你可以在该页面查看各个版本的 AT 固件、发布说明、配套文档及相应获取方式。

5.2 我该选择哪个版本？

请阅读 [AT 软件方案选型](#)。

5.3 版本管理

5.3.1 ESP-AT 发布固件管理

ESP-AT 发布是针对芯片的，通常是指发布一个或者几个芯片的 AT 固件。ESP-AT 发布的 [AT 固件](#) 所支持的 AT 命令集通常是向后兼容的。这意味着，你可以将新版本的 AT 固件更新至旧版本的设备中。

ESP-AT 发布的固件采用了和 [语义版本管理方法](#) 类似的方式，即你可以从字面含义理解每个版本的差异。例如 v3.3.0.0，v 代表版本，其后为版本号，版本号的格式如下：

```
<major>.<minor>.<patch>.<custom>
```

其中：

- <major> 为主要版本。例如 v4.0.0.0 代表有重大更新，通常包括引入新的芯片支持、新特性、和问题修复。
- <minor> 为次要版本。例如 v3.3.0.0 代表有较大更新，通常包括新增特性、ESP-IDF 版本升级、和问题修复。
- <patch> 为修复版本，也叫 bugfix 版本。例如 v2.4.2.0 代表仅修复了一些问题，并不增加任何新特性。
- <custom> 为自定义版本。通常用于下游代理商、或定制项目的版本。

5.3.2 ESP-AT 发布分支管理

ESP-AT 发布 AT 固件的同时，如果有主要版本或者次要版本的更新，会创建一个新的 [发布分支](#)。

ESP-AT 发布的分支采用了和 [语义版本管理方法](#) 类似的方式，即你可以从字面含义理解每个版本的差异。例如 `release/v3.3.0.0`，`release` 代表着发布的分支，`v` 代表着发布分支的版本，其后为版本号，版本号的格式如下：

```
<major>.<minor>.0.0
```

其中：

- `<major>` 为主要版本。例如 `release/v4.0.0.0` 代表有重大更新，通常包括引入新的芯片支持、新特性、和问题修复。
- `<minor>` 为次要版本。例如 `release/v3.3.0.0` 代表有较大更新，通常包括新增特性、ESP-IDF 版本升级、和问题修复。

通常情况下，待发布的 AT 固件会在发布分支上进行多轮测试，直到没有重大问题后，会发布 AT 固件并同步发布分支到 GitHub。

- 新的特性开发通常会在 `master` 分支上进行，不会合并到发布分支上。
- 问题修复通常会在 `master` 分支上进行，如果问题严重，会合并到发布分支上。

5.4 支持期限

ESP-AT 的每个主要版本和次要版本都有相应的支持期限。支持期限满后，版本停止更新维护，将不再提供支持。由于 ESP-AT 是基于 ESP-IDF 开发的项目，因此 ESP-AT 的支持期限是受限于 ESP-IDF 的支持期限。当前 ESP-AT 各版本信息如下：

AT 版本（发布时间）	发布固件适用芯片	IDF 版本	IDF 支持期限截止	AT 新版本迭代计划
v5.0.1.0 (2026.3.9)	• ESP32-C5 • ESP32-C61	~v5.5.3 (82e525ef)	2028.1.21	2027.8.21 - 2027.10.21
v5.0.0.0 (2025.11.24)	• ESP32-C5 • ESP32-C61	~v5.5.2 (cc569cbd)	2028.1.21	已发布新版本 v5.0.1.0
v4.2.0.0 (2026.8.7)	• ESP32 • ESP32-C2 • ESP32-C3 • ESP32-C6 • ESP32-S2	~v5.4.4 (296b6eab)	2027.7.5	2027.2.5 - 2027.4.5
v4.1.1.0 (2025.7.31)	• ESP32 • ESP32-C2 • ESP32-C3 • ESP32-C6 • ESP32-S2	~v5.4.2 (8ad0d3d8)	2027.7.5	已发布新版本 v4.2.0.0
v4.1.0.0 (2025.7.18)	• ESP32 • ESP32-C2 • ESP32-C3 • ESP32-C6	~v5.4.2 (8ad0d3d8)	2027.7.5	已发布新版本 v4.2.0.0
v4.0.0.0 (2023.12.29)	ESP32-C6	v5.1.2	2025.12.30	已发布新版本 v4.2.0.0
v3.4.0.0 (2024.6.7)	• ESP32 • ESP32-S2	v5.0.6	2025.5.29	已发布新版本 v4.2.0.0
v3.3.0.0 (2024.5.9)	• ESP32-C2 • ESP32-C3	v5.0.6	2025.5.29	已发布新版本 v4.2.0.0

ESP-AT 各发布版本支持期限如下：

- 支持期限的服务期
通常为从 AT 发布该芯片的 AT 固件开始，到计划发布该芯片下一个 AT 版本为止。下一个 AT 版本的发布时间通常在该芯片有重大问题需要修复，或者对应的 [ESP-IDF 支持期限](#) 结束前几个月（AT 发布说明中有介绍该芯片对应的 ESP-IDF 版本）。
- 支持期限的维护期
通常为从服务期结束后，到该芯片对应的 [ESP-IDF 支持期限](#) 结束（AT 发布说明中有介绍该芯片对应的 ESP-IDF 版本）。例如，ESP-IDF v5.0 的支持期限到 2025 年 5 月 29 日，那么 ESP-AT v3.0 ~ v3.3 的维护期限也到 2025 年 5 月 29 日。

一般而言：

- 一旦 AT 发布新的版本，则旧版本的支持期限的服务期结束，进入支持期限的维护期。
例如，AT 发布了 v3.3.0.0 版本（针对 ESP32-C2 和 ESP32-C3 芯片），那么 ESP32-C3 的 v3.2.0.0 版本的支持期限的服务期结束，进入支持期限的维护期；ESP32-C2 的 v3.1.0.0 版本的支持期限的服务期结束，进入支持期限的维护期。
- 如你有 GitHub 账号，请[订阅 AT 版本发布](#)，GitHub 将会在新版本发布的时候通知你。当你所使用的 AT 固件有 Bugfix 版本发布时，请做好升级至该 Bugfix 版本的规划。
- 请确保你所使用的版本停止更新维护前，已做好升级至新版本的规划。

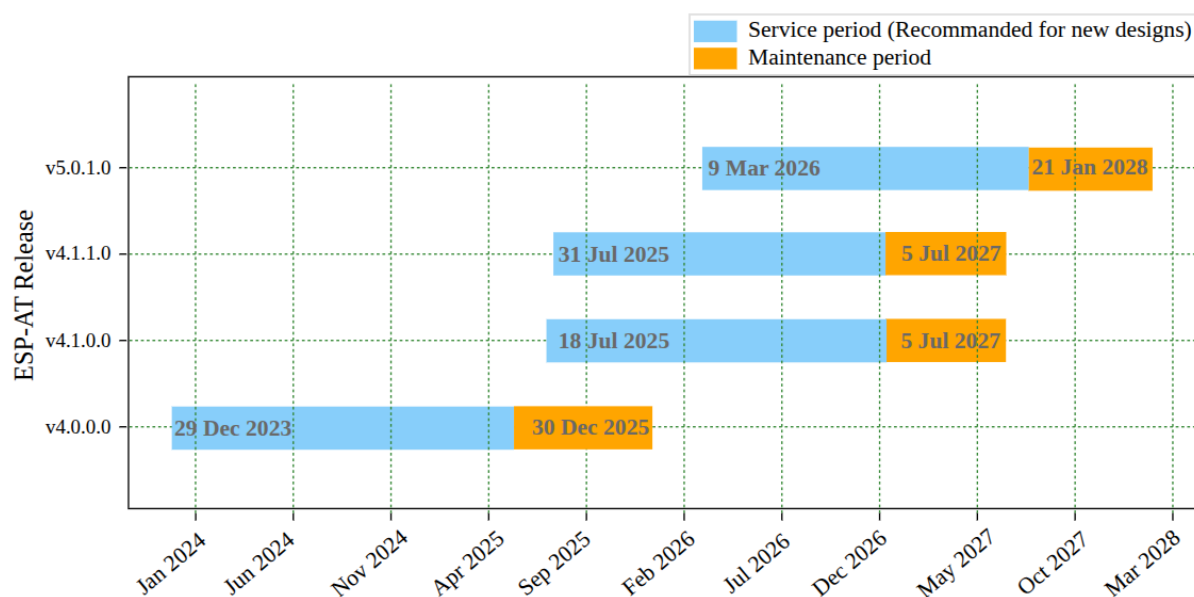


图 1: ESP-AT 版本支持期限

- 在支持期限内意味着 ESP-AT 团队将继续在 GitHub 的发布分支上进行重要 bug 修复、安全修复等，并根据需要定期发布新的 Bugfix 版本。

5.5 查看当前 AT 固件版本

请发送 `AT+GMR` 命令查看 AT 固件版本信息，参考 `AT+GMR` 命令下的参数说明了解更多信息。

5.6 订阅 AT 版本发布

- 第一步：登录你的 [GitHub](#) 账号
- 第二步：选择定制化的通知
- 第三步：定制发布应用

5.6.1 第一步：登录你的 GitHub 账号

在开始之前，请先 [登录你的 GitHub 账号](#)，因为订阅发布需要登录权限。

5.6.2 第二步：选择定制化的通知

访问 [ESP-AT 仓库](#)，点击页面右上角的 Watch，再点击 Custom。

5.6.3 第三步：定制发布应用

勾选 Releases 并点击 Apply。

这样就完成了订阅 AT 发布的操作。当有新的 AT 版本发布时，你将会收到 GitHub 的通知。

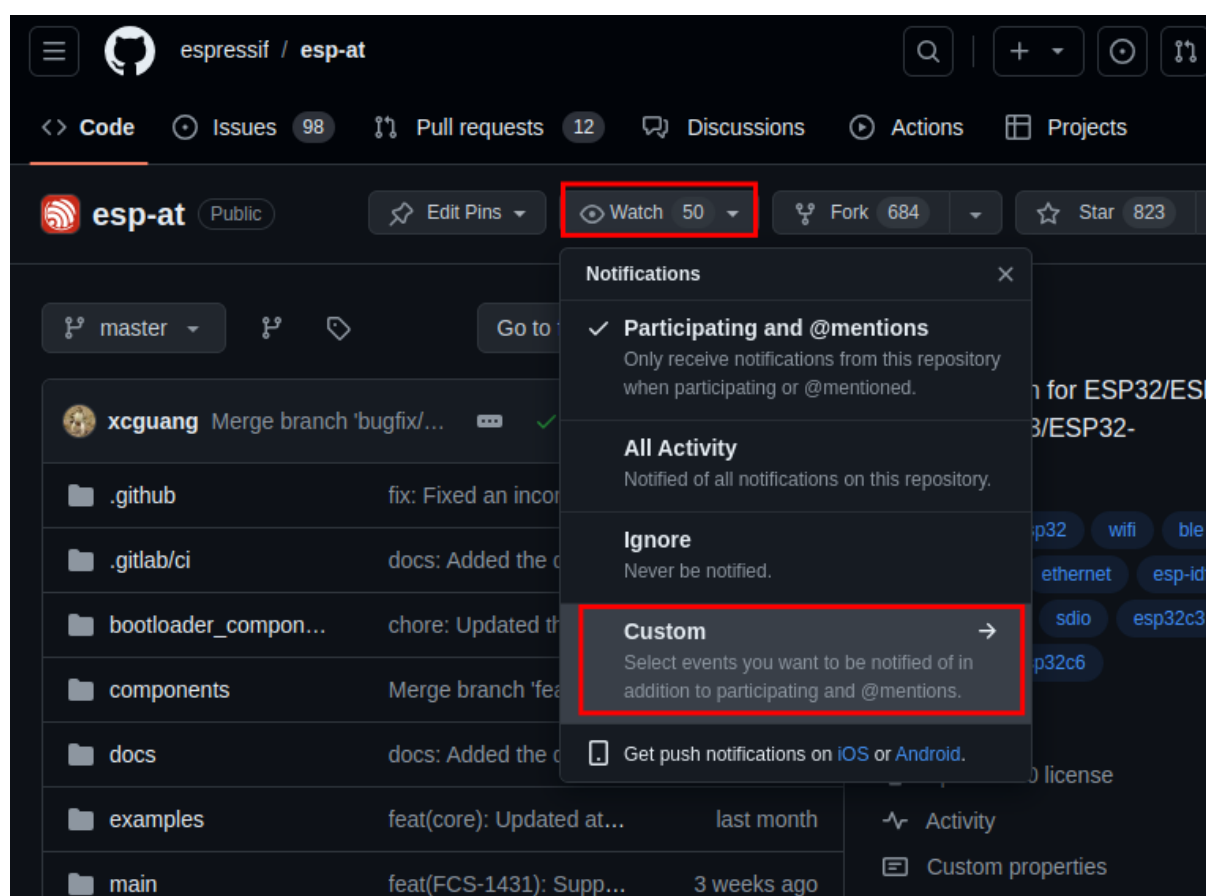


图 2: 定制化通知 (点击放大)

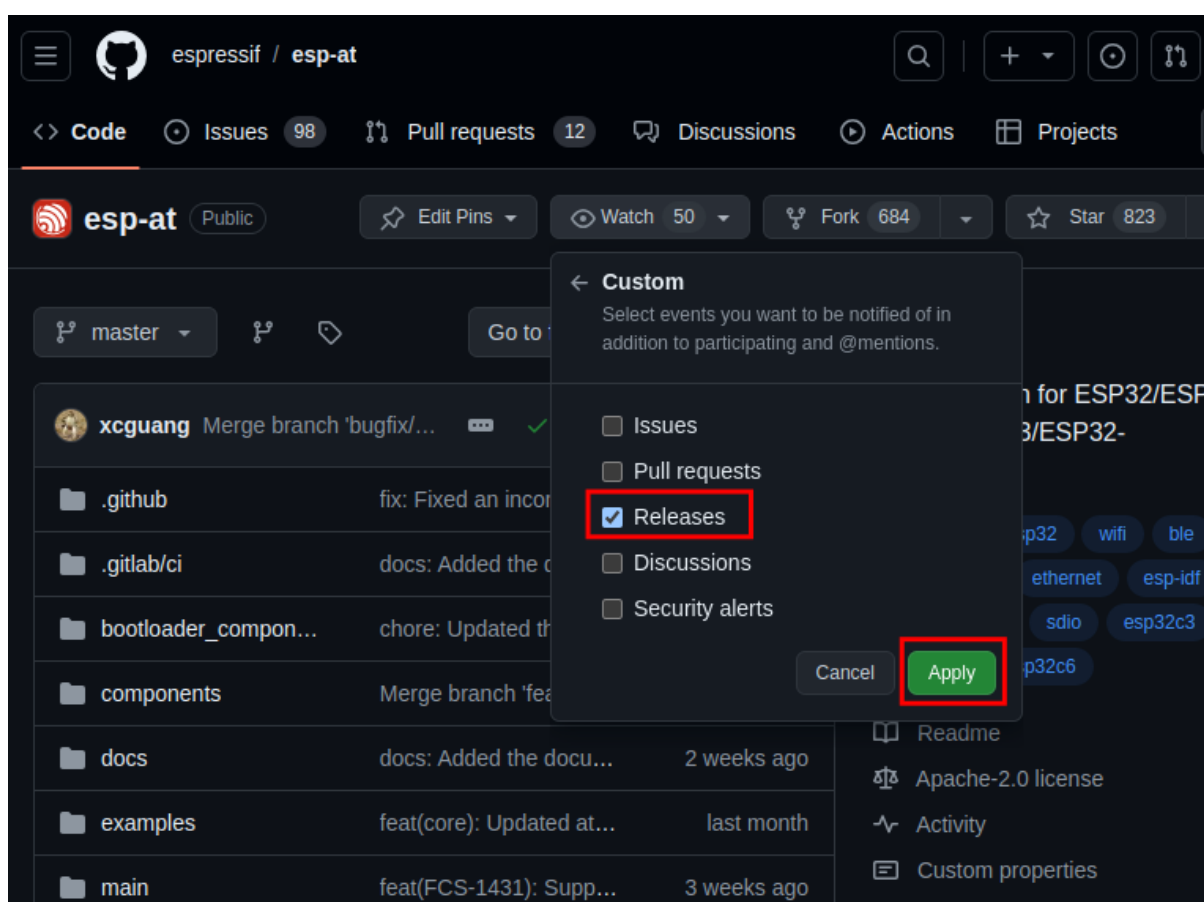


图 3: 定制发布应用（点击放大）

Chapter 6

如何编译和开发自己的 AT 工程

6.1 本地编译 ESP-AT 工程

本文档详细介绍了如何本地编译 ESP-AT 工程，并将生成的固件烧录到 ESP32-C6 设备中。当默认的[官方发布的固件](#)无法满足需求时，如你需要自定义[AT 端口管脚](#)、[低功耗蓝牙服务](#)以及[分区](#)等，那么就需要编译 ESP-AT 工程。

如果本地编译 ESP-AT 工程有困难，或者需要修改的代码量较少，推荐你通过[如何在 GitHub 网页上编译 ESP-AT 工程](#)。

6.1.1 详细步骤

请根据下方详细步骤，完成 ESP-AT 工程的克隆、环境安装、配置、编译以及烧录。**推荐你优先选择 Linux 系统开发。**

- 第一步：[ESP-IDF 快速入门](#)
- 第二步：获取 [ESP-AT](#)
- 第三步：安装环境
- 第四步：连接设备
- 第五步：配置工程
- 第六步：编译工程
- 第七步：烧录到设备
- [build.py 进阶用法](#)

6.1.2 第一步：ESP-IDF 快速入门

在编译 ESP-AT 工程之前，请先学习使用 ESP-IDF，因为 ESP-AT 是基于 ESP-IDF 开发的。

请你根据 [ESP-IDF v5.4 快速入门文档](#) 的指导，完成 hello_world 工程的配置、编译以及下载固件至 ESP32-C6 开发板等步骤。

备注：此步骤不是必须的，但如果你是初学者，强烈建议你完成此步骤，以熟悉 ESP-IDF 并确保顺利进行以下步骤。

完成上一步的 ESP-IDF 快速入门后，便可以开始下面的 ESP-AT 工程的编译。

6.1.3 第二步：获取 ESP-AT

编译 ESP-AT 工程需下载乐鑫提供的软件库文件 ESP-AT 仓库。

打开终端，切换到你要保存 ESP-AT 的工作目录，使用 `git clone` 命令克隆远程仓库，获取 ESP-AT 的本地副本。以下是不同操作系统的获取方式。

- Linux 或 macOS

```
cd ~/esp
git clone --recursive https://github.com/espressif/esp-at.git
```

- Windows

对于 ESP32-C6 系列模组，推荐你以管理员权限运行 [ESP-IDF 5.4 CMD](#)。

```
cd %userprofile%\esp
git clone --recursive https://github.com/espressif/esp-at.git
```

如果你在中国或无法访问 GitHub 的地区，可以通过以下镜像更快地克隆 ESP-AT：`git clone https://jihulab.com/esp-mirror/espressif/esp-at.git`。

ESP-AT 将下载至 Linux 和 macOS 的 `~/esp/esp-at`、Windows 的 `%userprofile%\esp\esp-at`。

备注：在本文档中，Linux 和 macOS 操作系统中 ESP-AT 的默认安装路径为 `~/esp`；Windows 操作系统的默认路径为 `%userprofile%\esp`。你也可以将 ESP-AT 安装在任何其它路径下，但请注意在使用命令行时进行相应替换。注意，ESP-AT 不支持带有空格的路径。

6.1.4 第三步：安装环境

运行项目工具 `install` 来安装环境。此安装工具将自动安装依赖的 Python 包、ESP-IDF 仓库以及 ESP-IDF 依赖的编译器、工具等。

- Linux 或 macOS

```
./build.py install
```

- Windows

```
python build.py install
```

如果是第一次安装环境，请为 ESP32-C6 设备选择以下配置选项。

- 选择 Platform name，例如 ESP32-C6 系列设备选择 `PLATFORM_ESP32C6`。Platform name 由 [factory_param_data.csv](#) 定义。
- 选择 Module name，例如 ESP32-C6-MINI-1 模组选择 `ESP32C6-4MB`。Module name 由 [factory_param_data.csv](#) 定义。
- 在选择启用或禁用 `silence mode` 之前，请先阅读[文档](#)，了解 `silence mode`。一般情况下请禁用。
- 如果 `build/module_info.json` 文件存在，上述三个配置选项将不会出现。因此，如果你想重新配置模组信息，请删除该文件。

以设置 Platform name 为 `ESP32C6`，Module name 为 `ESP32C6-4MB`，并禁用 `silence mode` 为例：

```
$ ./build.py install
Ready to install ESP-IDF prerequisites..

... (more lines of install ESP-IDF prerequisites)

Ready to install ESP-AT prerequisites..
```

(续下页)

(接上页)

```

... (more lines of install ESP-IDF prerequisites)

Platform name:
...
5. PLATFORM_ESP32C6
...
choose(range[1,7]):5

Module name:
1. ESP32C6-4MB (Firmware description: ...)
choose(range[1,1]):1

Enable silence mode to remove some logs and reduce the firmware size?
0. No
1. Yes
choose(range[0,1]):0
Platform name:ESP32C6 Module name:ESP32C6-4MB Silence:0

Cloning into 'esp-idf'...

... (more lines of clone esp-idf)

Ready to set up ESP-IDF tools..

... (more lines of set up ESP-IDF tools)

All done! You can now run:

./build.py build

```

6.1.5 第四步：连接设备

使用 USB 线将你的 ESP32-C6 设备连接到 PC 上，以下载固件和输出日志，详情请见[硬件连接](#)。注意，如果你在编译过程中不发送 AT 命令和接收 AT 响应，则不需要建立“AT 命令/响应”连接。关于更改默认端口管脚的信息请参考[如何设置 AT 端口管脚](#)。

6.1.6 第五步：配置工程

运行项目工具 menuconfig 来配置。

- Linux 或 macOS

```
./build.py menuconfig
```

- Windows

```
python build.py menuconfig
```

如果以上所有步骤都正确，则会弹出下面的菜单：

此菜单可以用来配置工程，如更改 AT 端口管脚、启用经典蓝牙功能等，如果不修改配置，则按照默认配置编译工程。

6.1.7 第六步：编译工程

运行以下命令编译工程。

- Linux 或 macOS

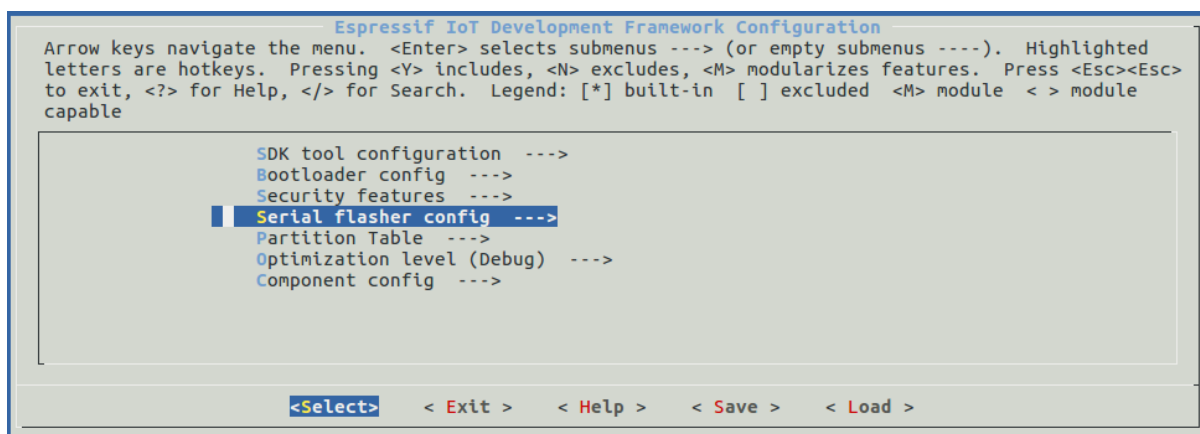


图 1: 工程配置 - 主窗口

```
./build.py build
```

- Windows

```
python build.py build
```

如果启用了蓝牙功能，固件尺寸会大大增加。请确保它不超过 ota 分区的大小。

编译完成后会在 build/factory 路径下生成打包好的量产固件。更多信息请参见[ESP-AT 固件差异](#)。

6.1.8 第七步：烧录到设备

运行以下命令将生成的固件烧录到 ESP32-C6 设备上。

- Linux 或 macOS

```
./build.py -p (PORT) flash
```

- Windows

```
python build.py -p (PORT) flash
```

注意请用 ESP32-C6 设备的串口名称替换 (PORT)。或者按照提示信息将固件烧录到 flash 中。仍然需要注意替换 (PORT)。

如果 ESP-AT bin 不能启动，并且打印出 “ota data partition invalid”，请运行 `python build.py erase_flash` 来擦除整个 flash，然后重新烧录 AT 固件。

6.1.9 build.py 进阶用法

build.py 脚本是基于 idf.py 封装的工具（即 `idf.py <cmd>` 功能均包含在 `build.py <cmd>` 里），你可以运行以下命令查看更多用法。

- Linux 或 macOS

```
./build.py --help
```

- Windows

```
python build.py --help
```

6.2 网页编译 ESP-AT 工程

本文档详细介绍了如何通过网页编译 ESP-AT 工程。当默认的[官方发布的固件](#)无法满足需求时，如你需要开启 [WebSocket 功能](#) 并关闭 [mDNS 功能](#)，那么就需要编译 ESP-AT 工程。通常推荐你[本地搭建环境编译 ESP-AT 工程](#)，但如果你本地搭建环境编译有困难，则请参考本文档，通过网页编译 ESP-AT 工程。

注意：网页编译的 AT 固件，需要你根据自己的产品自行测试验证功能。

请保存好下载的固件以及下载链接，用于后续可能的问题调试。

6.2.1 详细步骤

请根据下方详细步骤，完成 ESP-AT 工程的 Fork、环境配置、代码修改以及编译。

- 第一步：登录你的 [GitHub 账号](#)
- 第二步：Fork [ESP-AT 工程](#)
- 第三步：开启 [GitHub Actions 功能](#)
- 第四步：配置编译 [ESP-AT 工程](#) 所需的密钥
- 第五步：使用 [github.dev](#) 编辑器修改和提交代码
- 第六步：[GitHub Actions](#) 编译 AT 固件

6.2.2 第一步：登录你的 GitHub 账号

在开始之前，请先 [登录你的 GitHub 账号](#)，因为编译和下载固件需要登录权限。

6.2.3 第二步：Fork ESP-AT 工程

访问 [ESP-AT 仓库](#)，点击页面右上角的 Fork。

点击 Create fork 复制仓库到自己的 GitHub 账号下。默认的 Fork 仅拷贝 master 分支。如果你需要基于指定的分支修改代码，请取消勾选 Copy the master branch only。

6.2.4 第三步：开启 GitHub Actions 功能

点击仓库下的 Actions 进入 GitHub Actions 页面。

点击 I understand my workflows, go ahead and enable them 启用 GitHub Actions。

启用 GitHub Actions 完成。

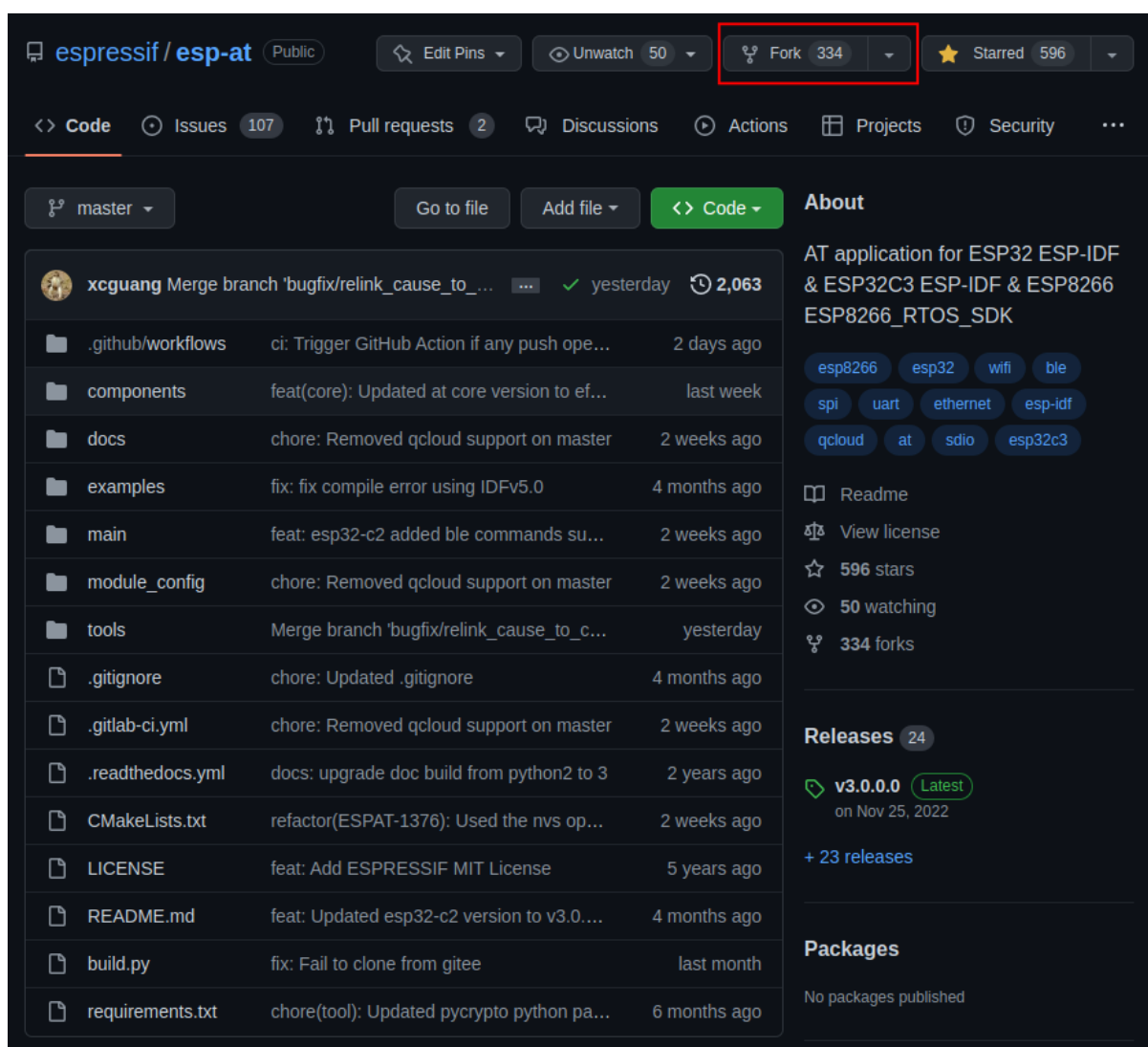


图 2: 点击 Fork

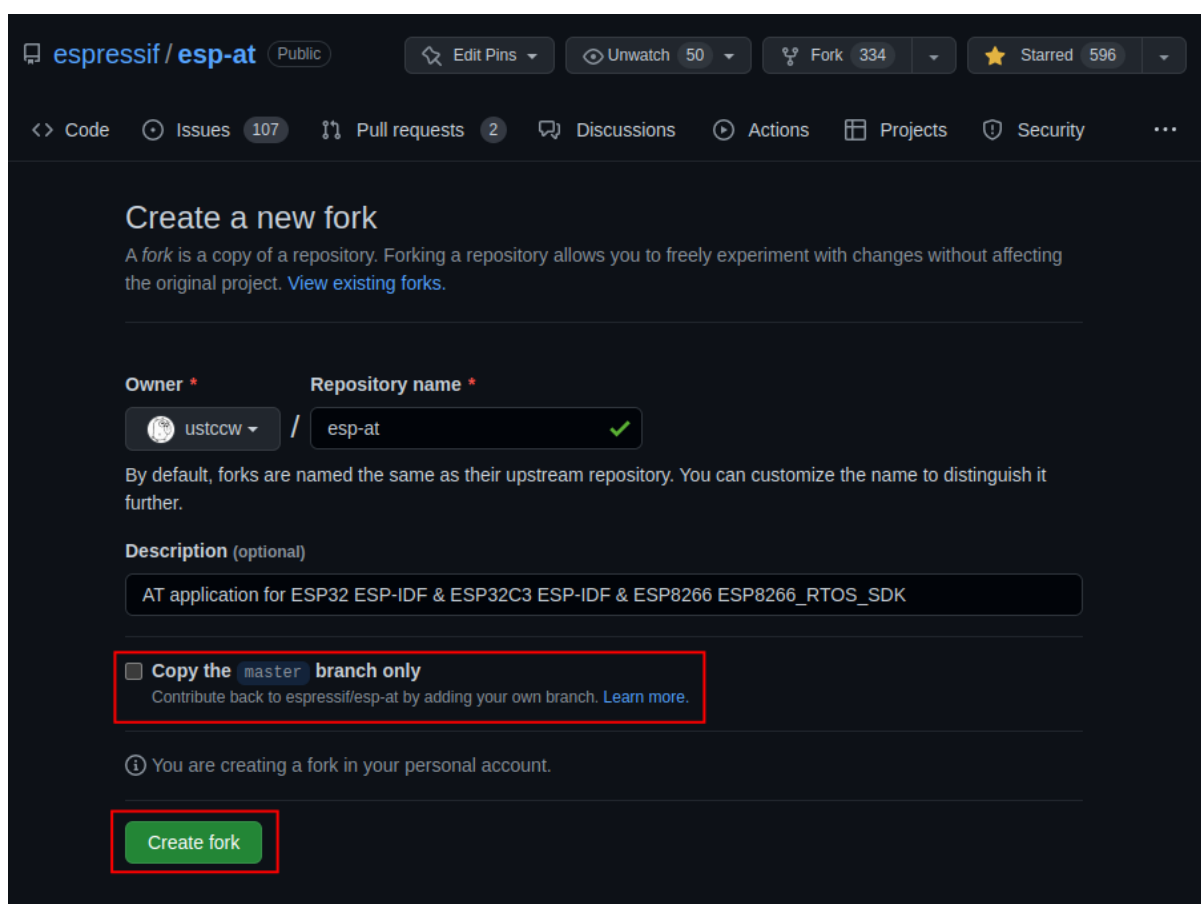


图 3: 创建 Fork

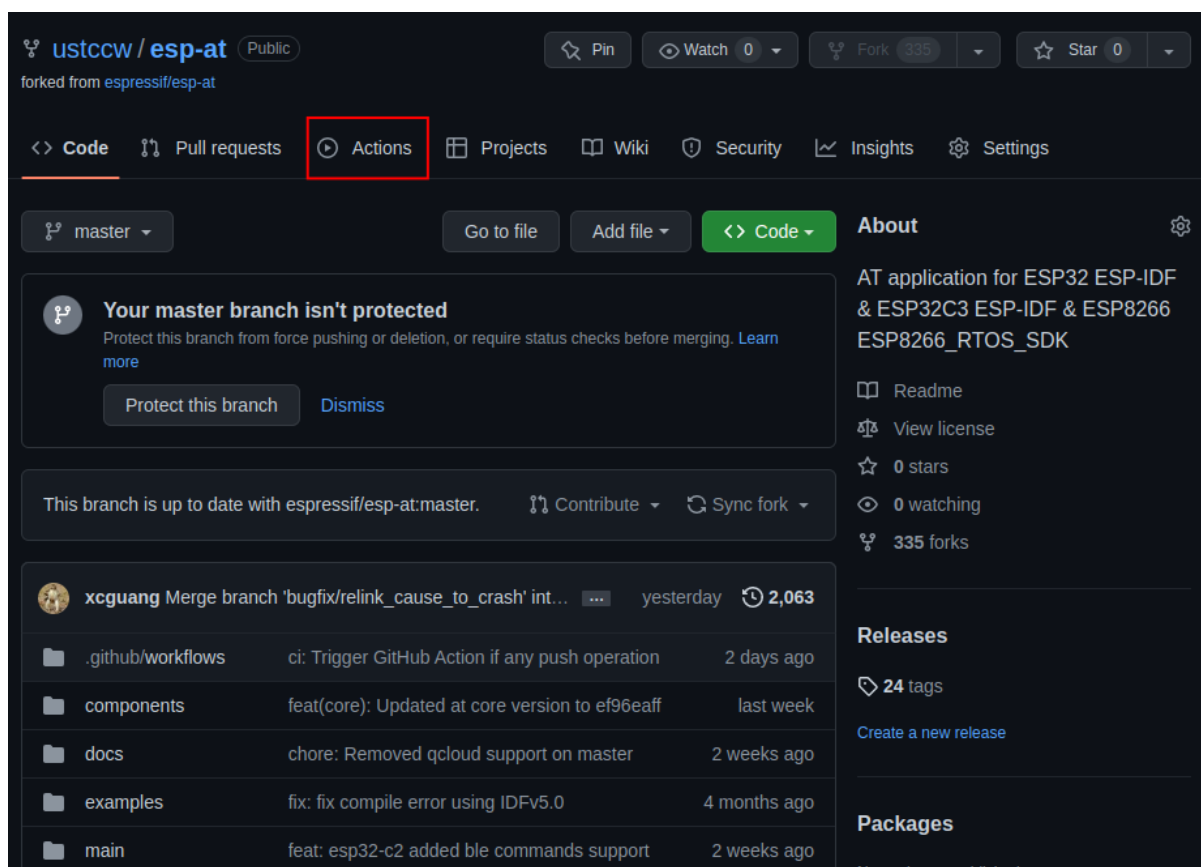


图 4: 点击 Actions

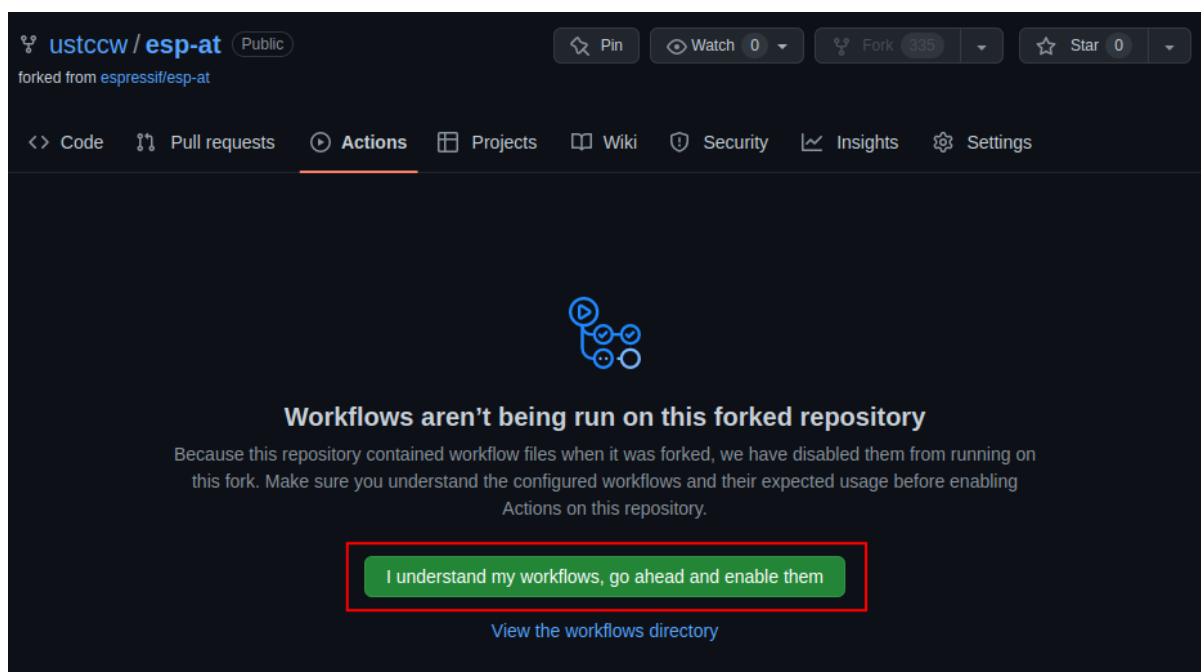


图 5: 启用 GitHub Actions

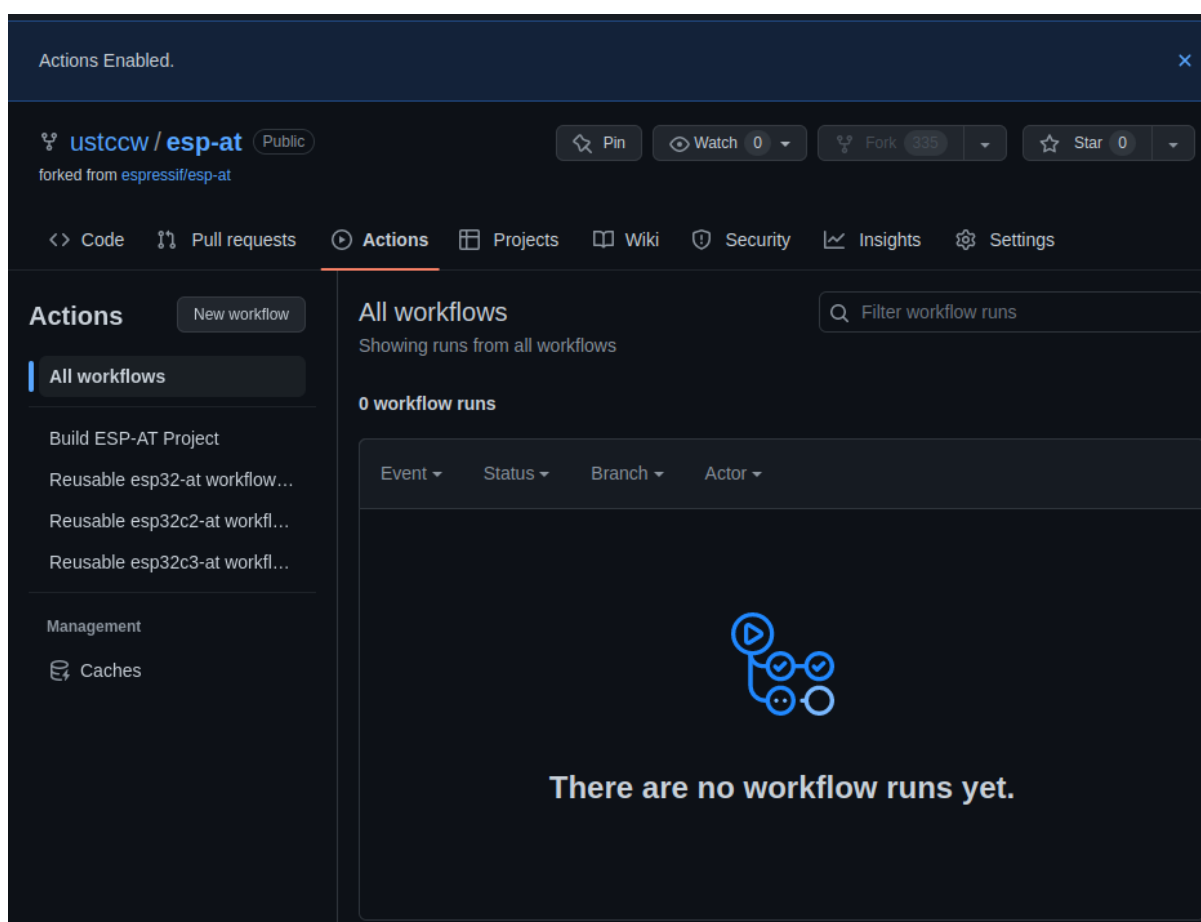


图 6: 启用 GitHub Actions 完成

6.2.5 第四步：配置编译 ESP-AT 工程所需的密钥

如果你有 [Espressif OTA server](#) 的账户和 OTA token，并且需要使用 `AT+CIUPDATE` 命令升级 AT 固件，那么需要完成此步骤。否则，建议你禁用 `CONFIG_AT_OTA_SUPPORT`（有关更多详细信息，请参阅[第五步：使用 github.dev 编辑器修改和提交代码](#)）或者跳过此步骤。

点击仓库下的 Settings 进入设置页面。

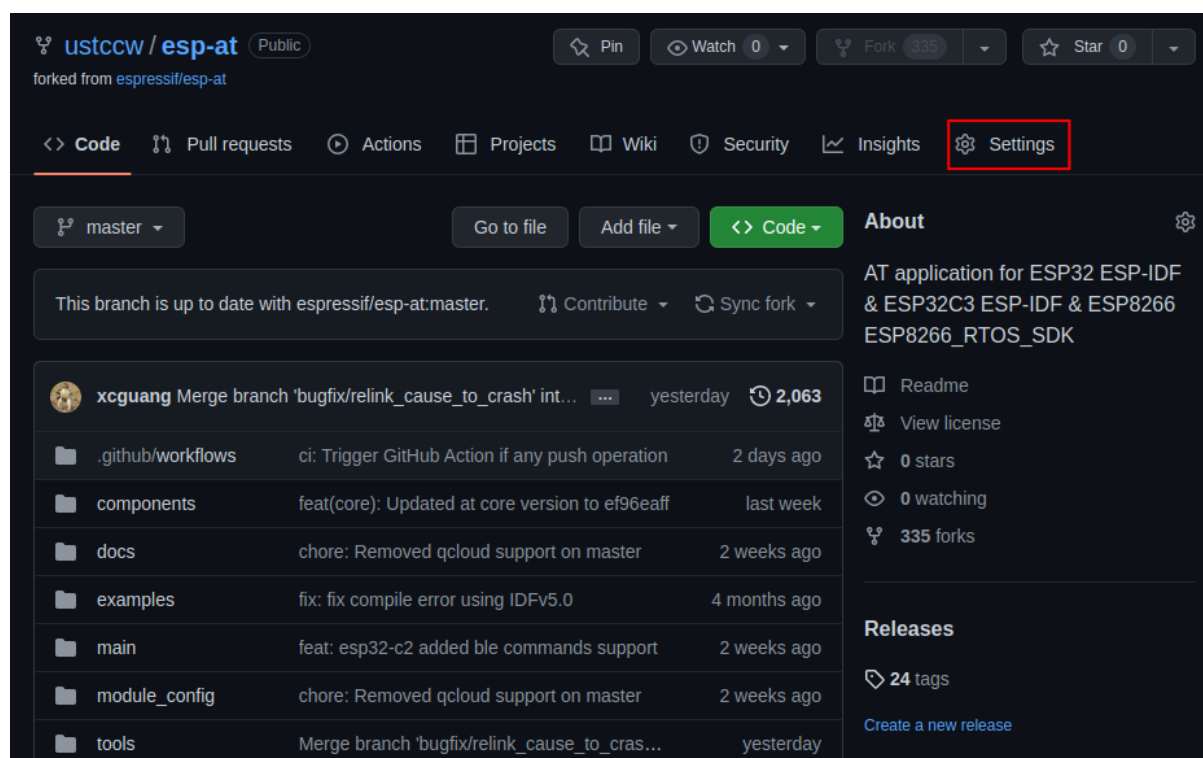


图 7: 点击 Settings

点击 Settings > Secrets and variables > Actions 进入 Action 配置页面。

点击 New repository secret 进入密钥创建页面。

输入 Name 和 Secrets，点击 Add secret 创建一个新的密钥。此处的密钥即是 OTA token。

需要配置的可能的密钥名称如下：

```
AT_OTA_TOKEN_WROOM32
AT_OTA_TOKEN_WROVER32
AT_OTA_TOKEN_ESP32_MINI_1
AT_OTA_TOKEN_ESP32_PICO_D4
AT_OTA_TOKEN_ESP32_SOLO_1
AT_OTA_TOKEN_ESP32C3_MINI
ESP32C2_2MB_TOKEN
ESP32C2_4MB_TOKEN
ESP32C5_4MB_TOKEN
ESP32C6_4MB_TOKEN
ESP32C61_4MB_TOKEN
```

如果需要更多的模组支持 AT 固件升级，请重复上面步骤，再次添加不同的密钥，所有密钥创建完成的页面如下。

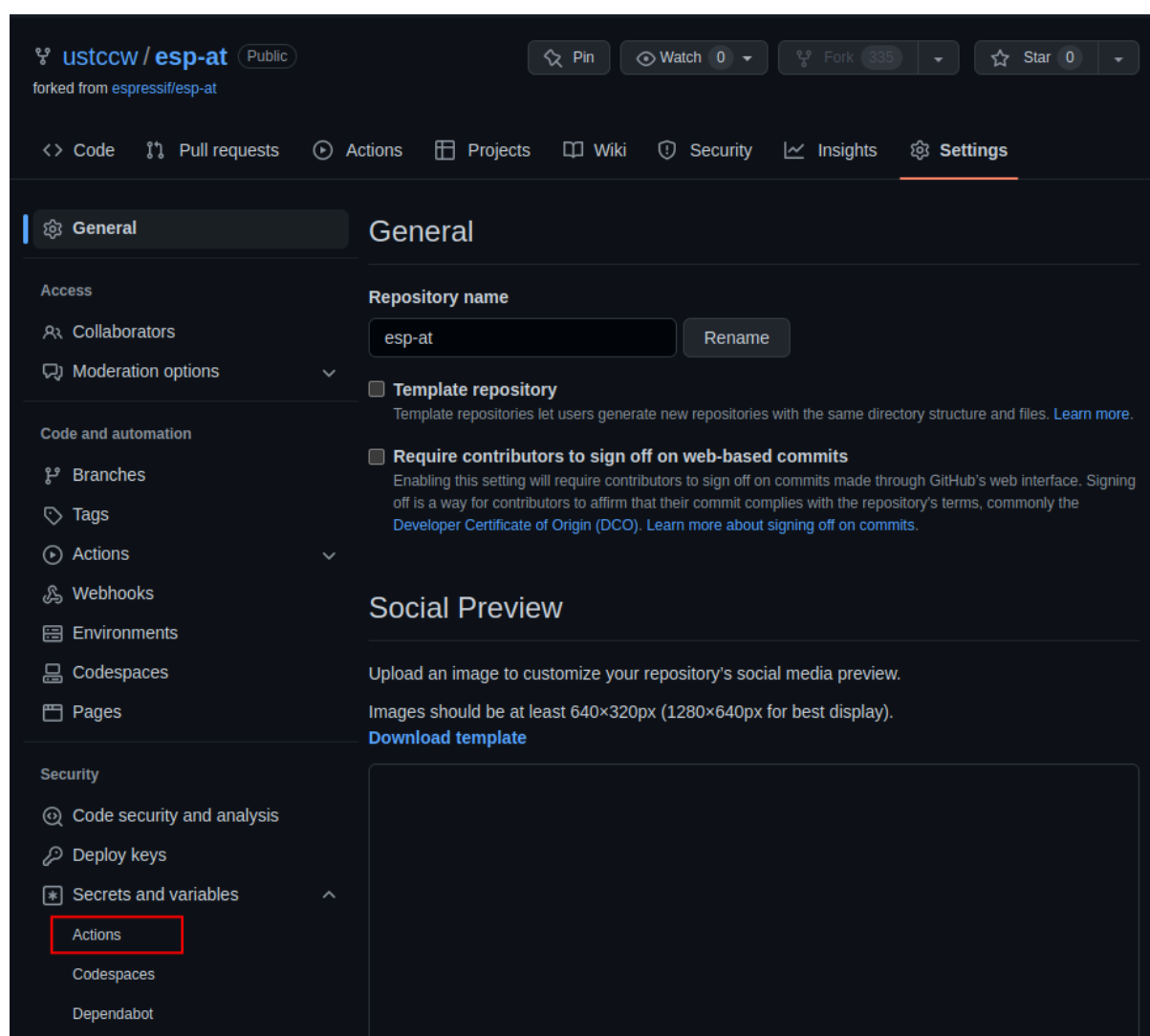


图 8: 进入 Actions 配置页面

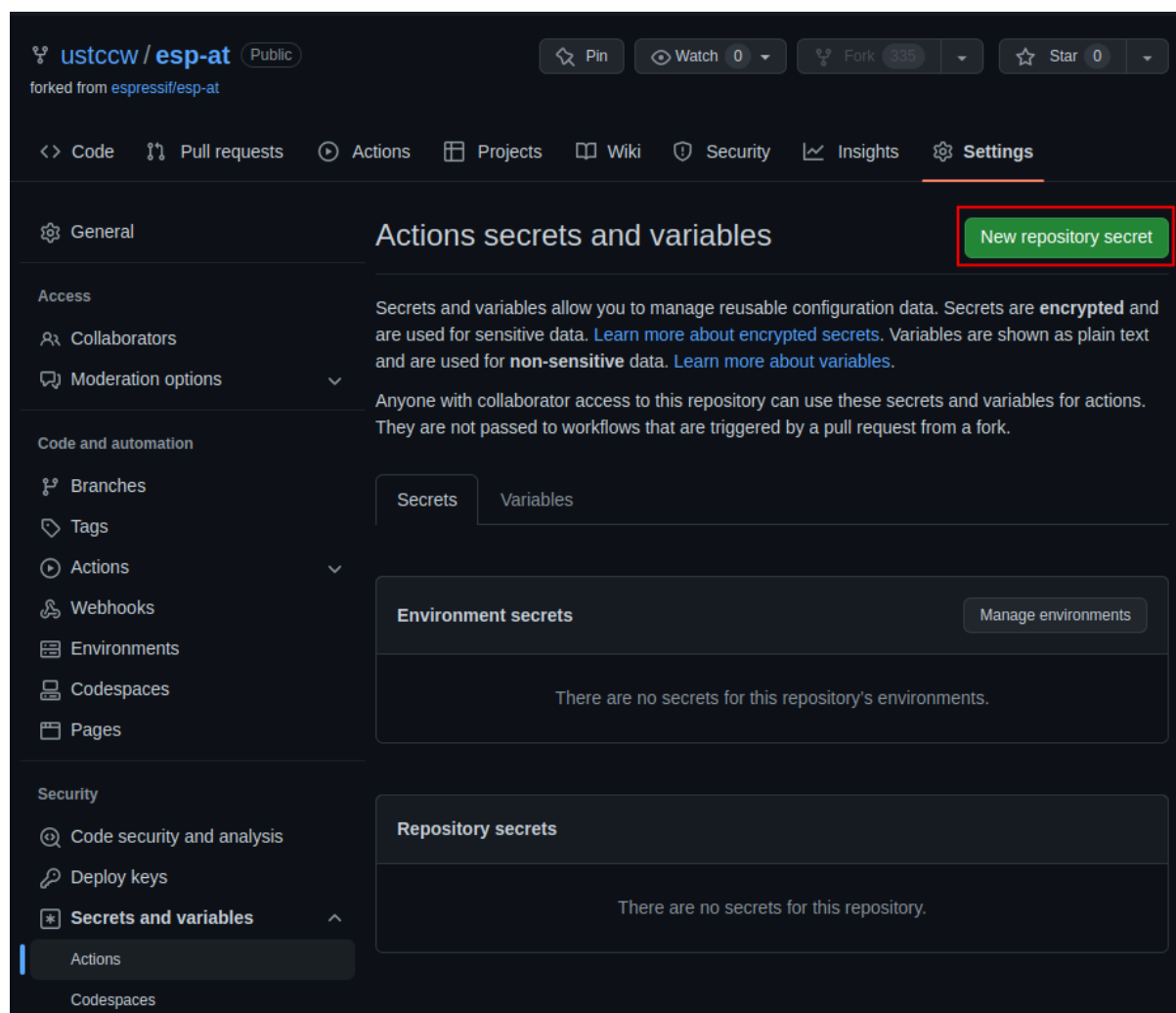


图 9: 创建密钥页面

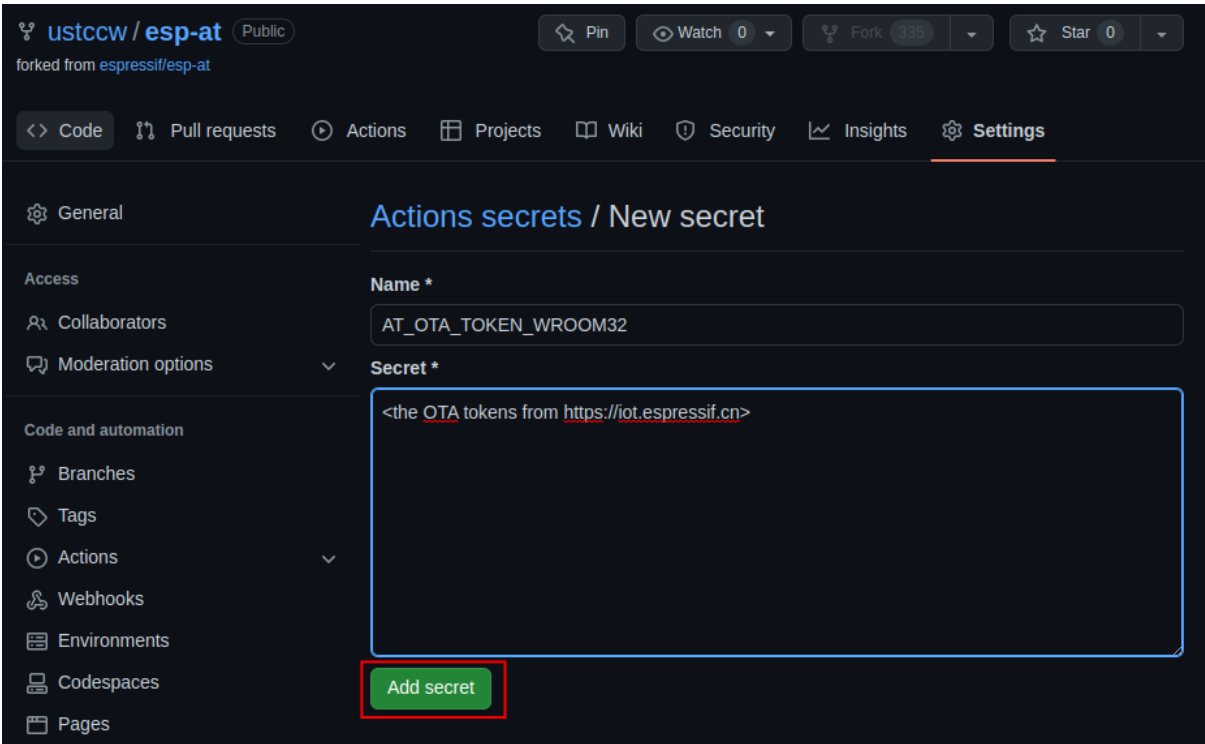


图 10: 创建密钥

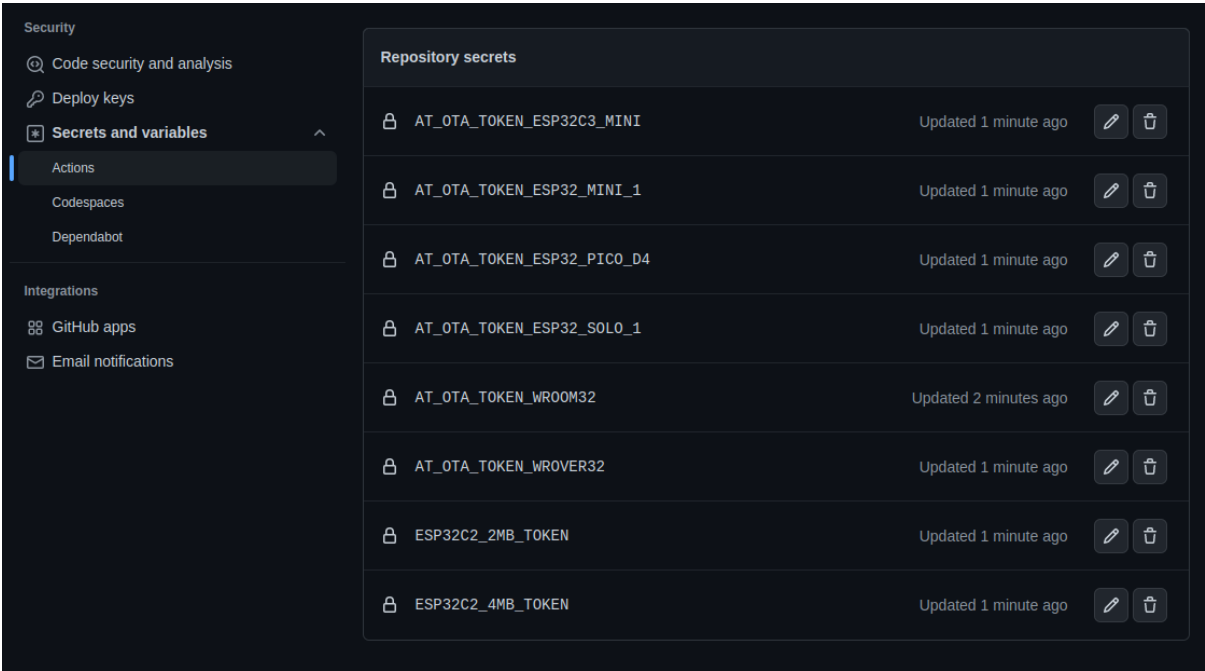


图 11: 创建密钥完成

6.2.6 第五步：使用 github.dev 编辑器修改和提交代码

第五步也可以参考 [github.dev](#) 官方文档，下面是示例过程。

5.1 打开 github.dev 编辑器

回到仓库主页，选择要修改的分支名称。

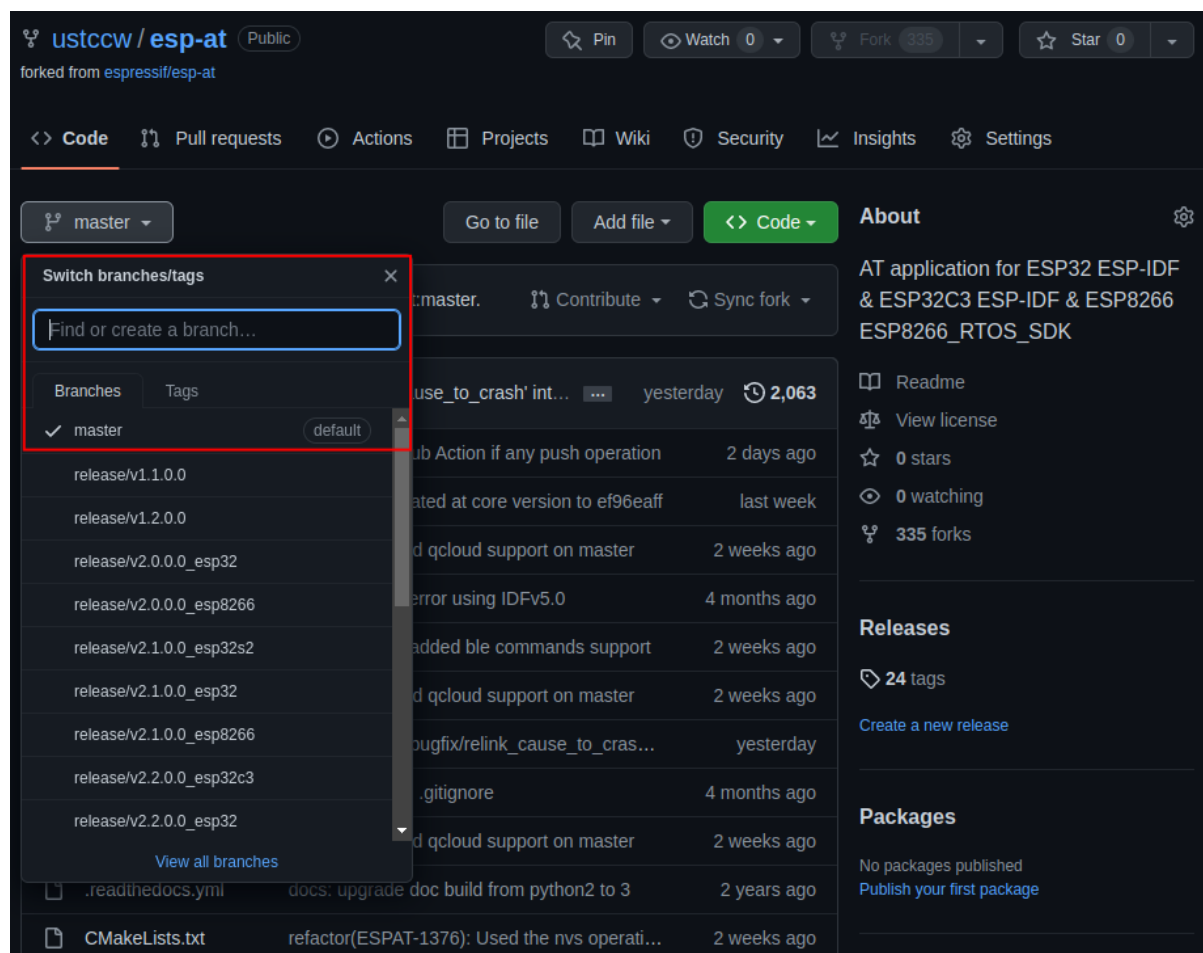


图 12: 选择分支

按键盘上点号 `.`，使用 [github.dev](#) 编辑器打开分支代码。

5.2 创建新分支

在编辑器的底部，单击状态栏中的分支名称，在下拉菜单中，单击要切换到的分支或输入新分支的名称，然后单击 创建新分支。

单击 Switch to Branch，创建分支。

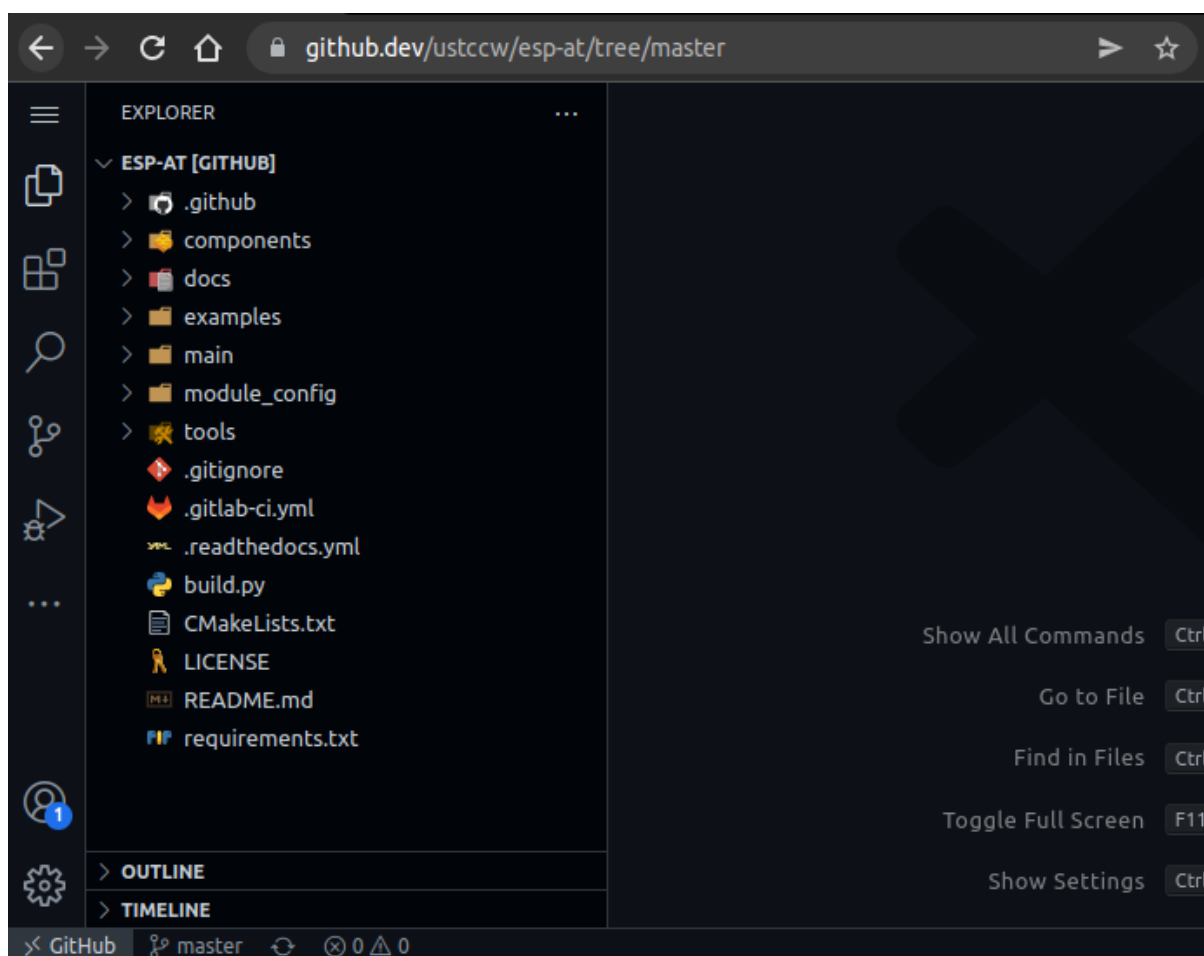


图 13: 打开分支代码

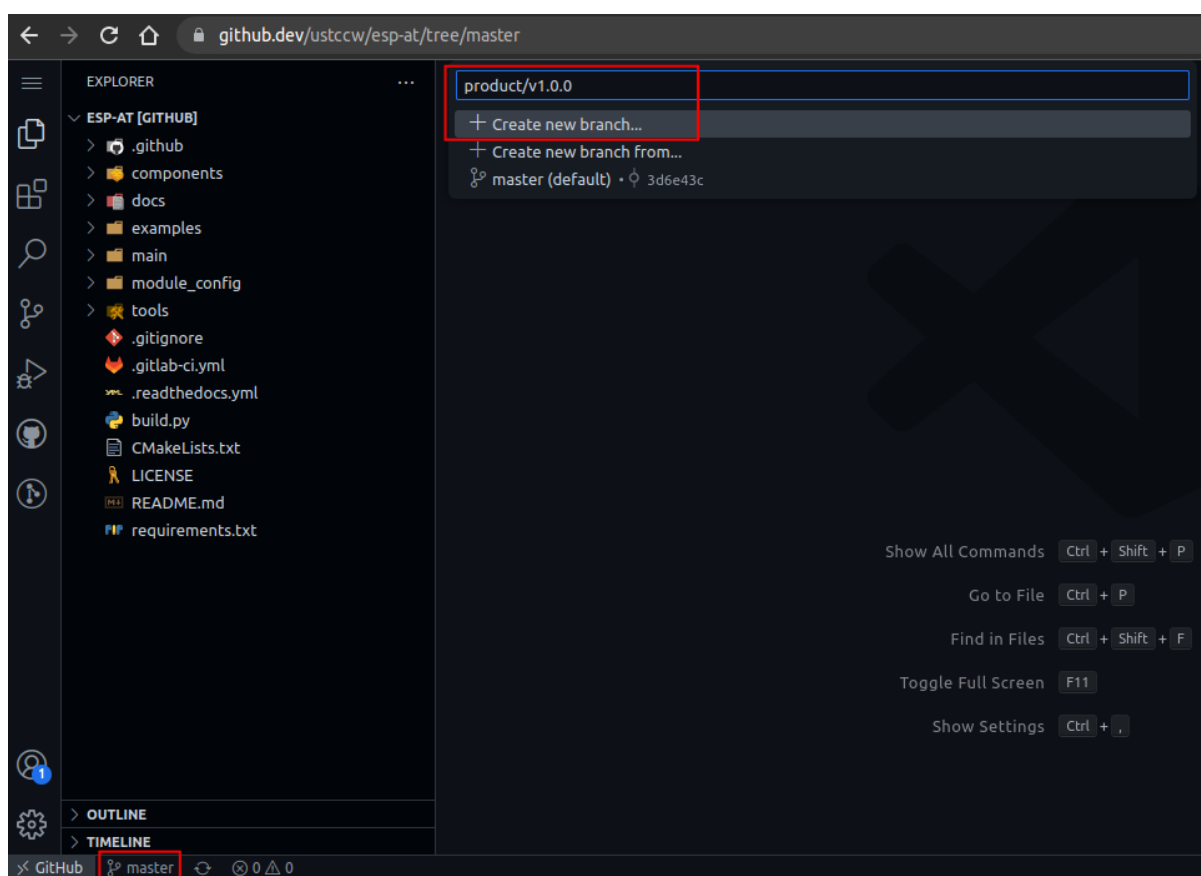


图 14: 创建新分支（点击放大）

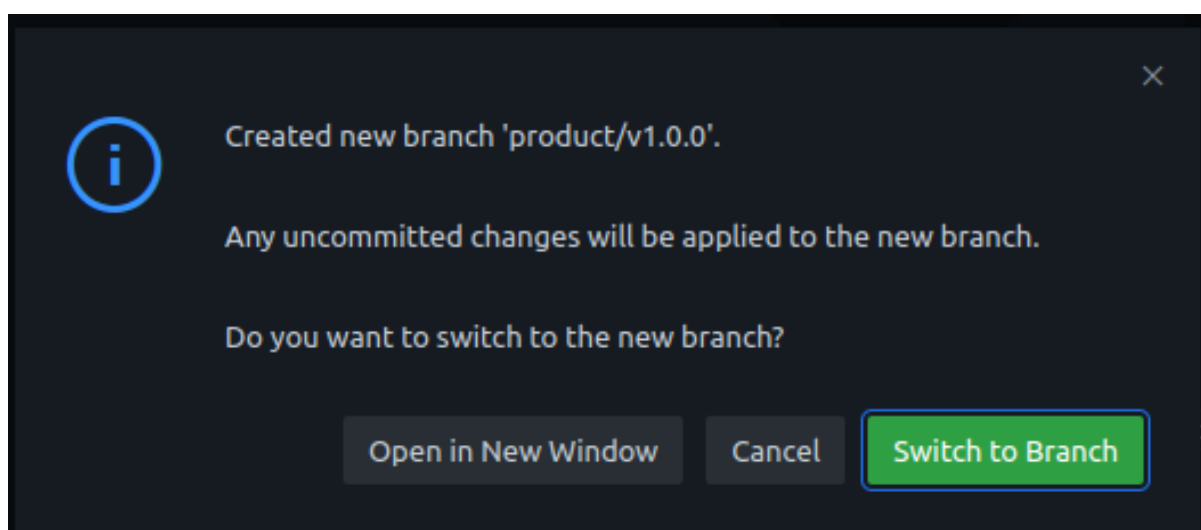


图 15: 确认创建分支

分支创建完成。

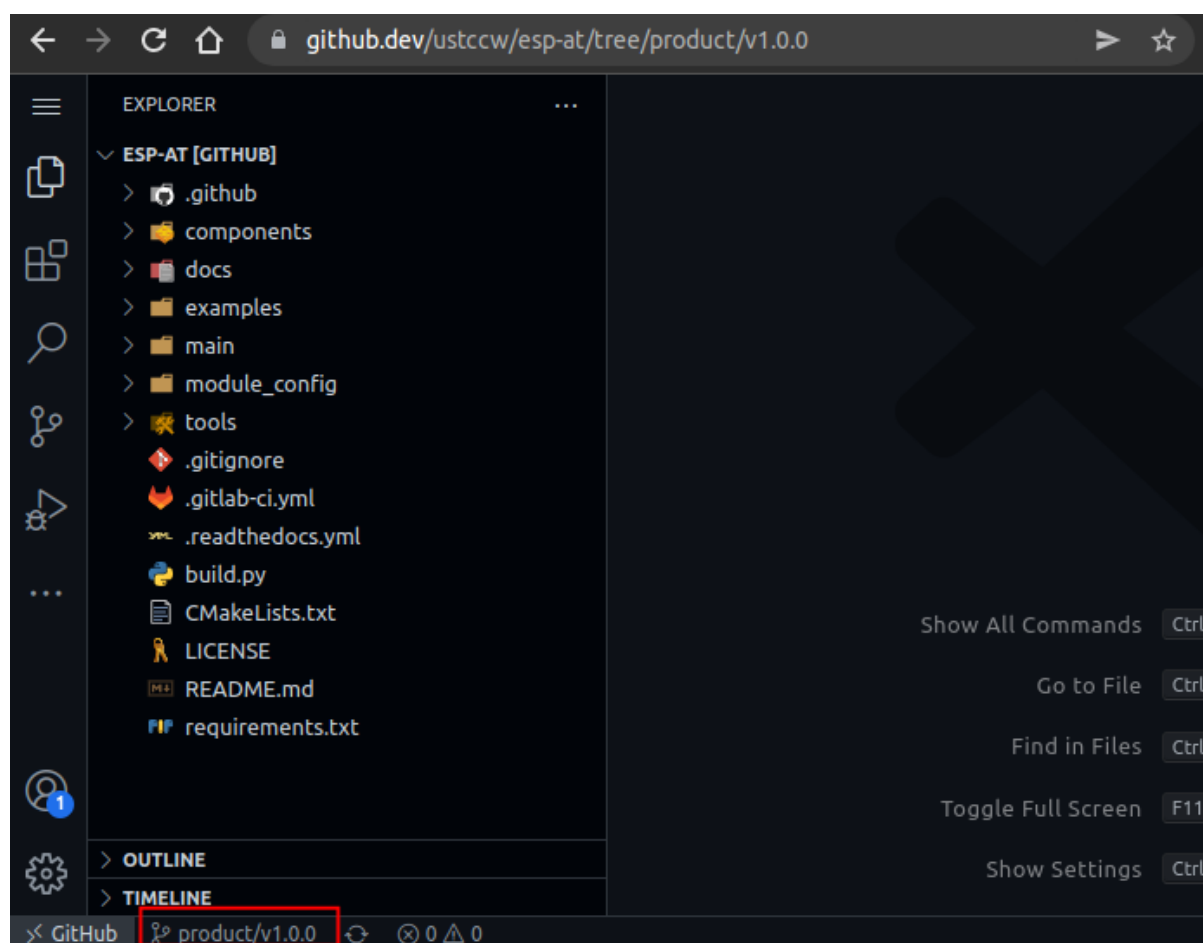


图 16: 分支创建完成

5.3 提交更改

在 github.dev 编辑器中修改代码。例如开启 [WebSocket 功能](#) 并关闭 [mDNS 功能](#)，则需要打开模块的配置文件 `esp-at/module_config/<your_module_name>/sdkconfig.defaults` 文件，增加如下行：

```
CONFIG_AT_WS_COMMAND_SUPPORT=y
CONFIG_AT_MDNS_COMMAND_SUPPORT=n
```

在活动栏中，单击 源代码管理视图。单击已更改文件旁边的加号 + 暂存修改。

输入提交消息，描述你所做的更改。点击 Commit & Push 提交。

备注： 如果你想要启用或禁用 AT 固件的 *silence mode*，请参考 [如何启用或禁用 silence mode](#) 文档。

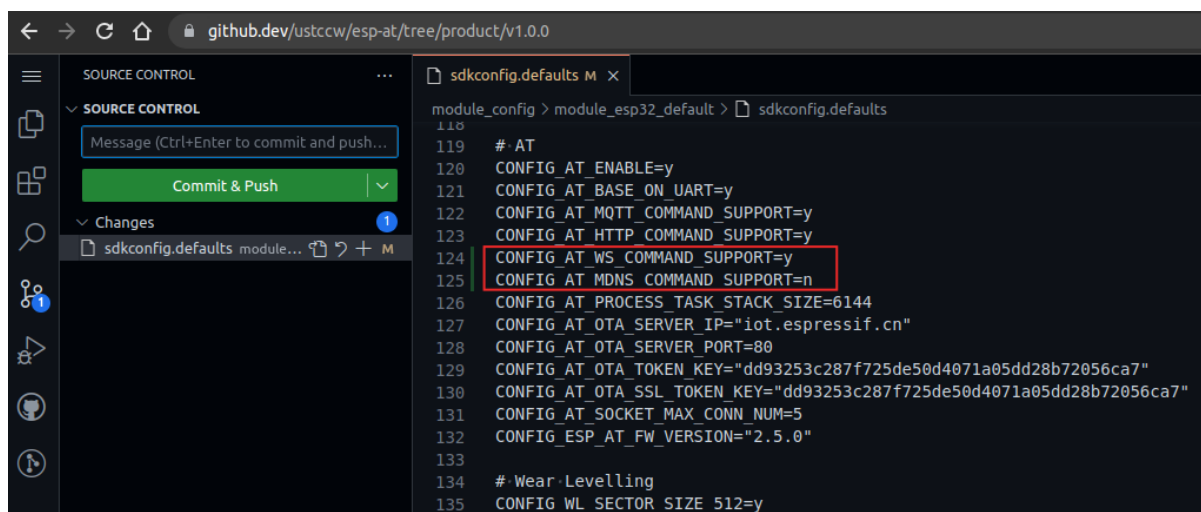


图 17: 增加 WebSocket 命令支持，禁用 mDNS 功能（点击放大）

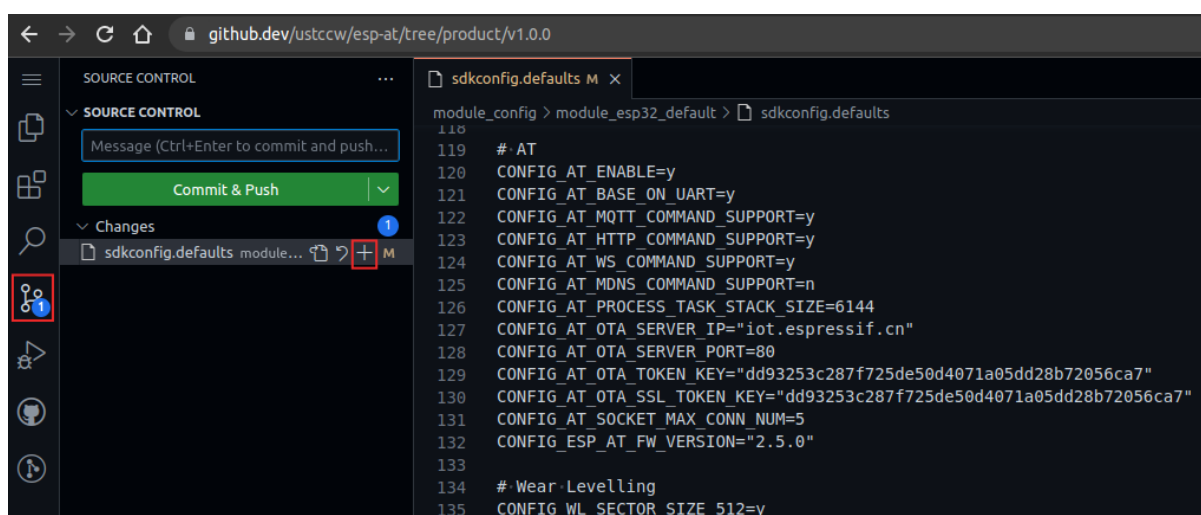


图 18: 暂存修改（点击放大）

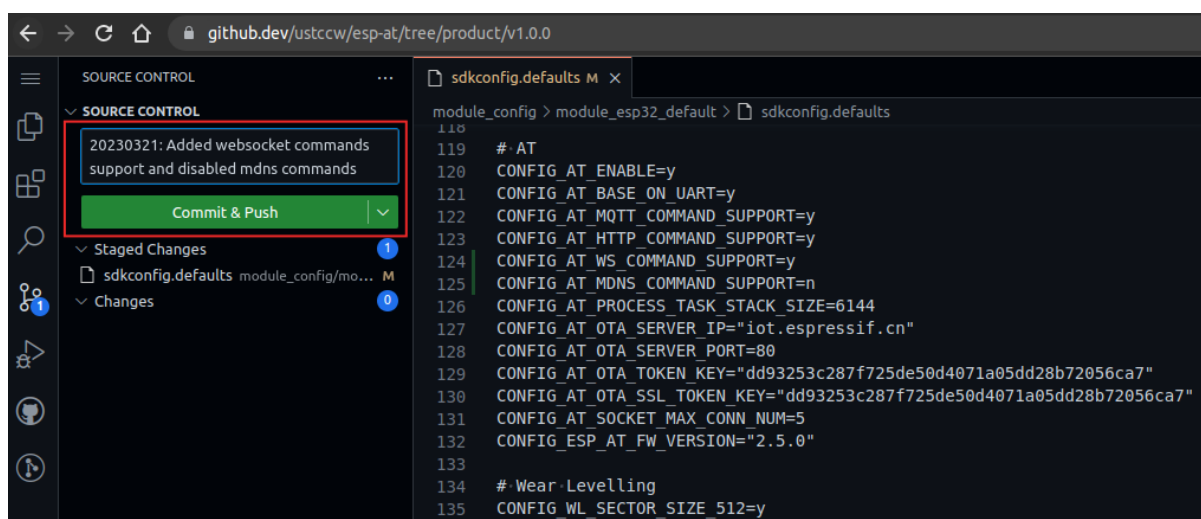


图 19: 提交修改（点击放大）

6.2.7 第六步：GitHub Actions 编译 AT 固件

上述步骤完成之后，会自动触发 GitHub Actions 编译你的 ESP-AT 固件。请参考[如何从 GitHub 下载最新临时版本 AT 固件](#) 文档，下载你所需要的 AT 固件（注意：文档里的步骤都是基于你自己账号下的 esp-at 仓库进行的，而不是 <https://github.com/espressif/esp-at> 仓库）。GitHub 上的 AT 固件会在到达 90 天有效期后自动失效，请自行保存下载的 AT 固件。

6.3 如何设置 AT 端口管脚

本文档介绍了如何修改 ESP32-C6 系列固件中的 *AT port* 管脚。默认情况下，ESP-AT 使用两个 UART 接口作为 AT 端口：一个用于输出日志（以下称为日志端口），另一个用于发送 AT 命令和接收响应（以下称为命令端口）。

如果你已经有 ESP32-C6 AT 固件，并且只需要修改命令端口管脚，请直接通过 *at.py* 工具修改固件，为你的模组生成新的固件。否则你要修改 ESP32-C6 设备的 AT 端口管脚，需要完成下面步骤：

- [克隆 ESP-AT 工程](#)。
- 在 menuconfig 配置工具或 factory_param_data.csv 表格中修改对应管脚。
- [编译工程](#)。
- 将新的 *bin* 文件烧录进设备。

本文档重点介绍如何修改管脚，点击上面的链接了解其它步骤的详细信息。

备注：使用其它接口作为 AT 命令接口请参考 [使用 AT SPI 接口](#)，[AT through SPI](#) 和 [使用 AT 套接字接口](#)。

6.3.1 ESP32-C6 系列

ESP32-C6 AT 固件的日志端口和命令端口管脚可以自定义为其它管脚，请参阅 [《ESP32-C6 技术参考手册》](#) 查看可使用的管脚。

修改日志端口管脚

默认情况下，乐鑫提供的 ESP32-C6 AT 固件使用以下 UART0 管脚输出日志：

- TX: GPIO16
- RX: GPIO17

在编译 ESP-AT 工程时，可使用 menuconfig 配置工具将其修改为其它管脚：

- `./build.py menuconfig > Component config > ESP System Settings > Channel for console output > Custom UART`
- `./build.py menuconfig > Component config > ESP System Settings > UART TX on GPIO#`
- `./build.py menuconfig > Component config > ESP System Settings > UART RX on GPIO#`

修改命令端口管脚

默认情况下，UART1 用于发送 AT 命令和接收 AT 响应，其管脚定义在 [factory_param_data.csv](#) 表格中的 `uart_port`、`uart_tx_pin`、`uart_rx_pin`、`uart_cts_pin` 和 `uart_rts_pin` 列。

你可以直接在 `factory_param_data.csv` 表中修改端口管脚：

- 打开你本地的 `factory_param_data.csv`。
- 找到模组所在的行。

- 根据需要设置 `uart_port`（如果希望 AT 日志口同时用作 AT 命令口，则需要修改此行，同时保证下面的 `uart_tx_pin` 和 `uart_rx_pin` 和 AT 日志口的管脚一样）。
- 根据需要设置 `uart_tx_pin` 和 `uart_rx_pin`（你需要保证将要修改的管脚，未被其它功能使用，包括 AT 日志口的管脚）。
- 若不需要使用硬件流控功能，请将 `uart_cts_pin` 和 `uart_rts_pin` 设置为 -1。
- 保存表格。

6.4 添加自定义 AT 命令

本文档介绍了如何添加自定义的 AT 命令，以 `at_custom_cmd` 为例，展示每个步骤的示例代码。

自定义基本的、功能良好的命令，请参考以下步骤。

- 第一步：定义 AT 命令
- 第二步：定义注册 AT 命令函数
- 第三步：增加组件依赖
- 第四步：增加链接选项
- 第五步：设置组件环境变量
- 第六步：编译 AT 固件

正确完成上述步骤后，请运行 `AT+TEST` 命令获取运行结果。

自定义相对复杂的命令，参考示例如下：

- 定义返回消息
- 获取命令参数
- 省略命令参数
- 阻塞命令的执行
- 从 AT 命令端口获取输入的数据

AT 命令集的源代码不开源，以 库文件 的形式呈现，它也是解析自定义的 AT 命令的基础。

6.4.1 自定义 AT 命令

第一步：定义 AT 命令

你可在 `at_custom_cmd.c` 和 `at_custom_cmd.h` 文件中定义 AT 命令，也可在 `examples/at_custom_cmd/custom` 和 `examples/at_custom_cmd/include` 目录下创建新的源文件和头文件定义 AT 命令。

在自定义 AT 命令之前，请先确定 AT 命令的名称和类型。

命令命名规则：

- 命令应以 + 符号开头。
- 支持字母 (A~Z, a~z)、数字 (0~9) 及其它一些字符 (!、%、-、.、/、:、_)，详情请见 [AT 命令分类](#)。

命令类型：

每条 AT 命令最多可以有四种类型：测试命令、查询命令、设置命令和执行命令，更多信息参见 [AT 命令分类](#)。

然后，定义所需类型的命令。假设 `AT+TEST` 支持所有的四种类型，下面是定义 AT 命令的名称和所需支持的类型，以及定义每种类型的示例代码。

- 首先，调用 `esp_at_cmd_t` 来定义 AT 命令的名称和所需支持的类型，下面的示例代码定义了名称为 `+TEST`（省略了 AT）并支持所有四种类型的命令。

```
static const esp_at_cmd_t at_custom_cmd[] = {
    {"+TEST", at_test_cmd_test, at_query_cmd_test, at_setup_cmd_test, ↵
    ↵ at_exe_cmd_test},
    /**
     * @brief 你可以在此处定义自己的 AT 命令
     */
};
```

备注： 如果不定义某个类型，将其设置为 NULL。

- 测试命令：

```
static uint8_t at_test_cmd_test(uint8_t *cmd_name)
{
    uint8_t buffer[64] = {0};
    snprintf((char *)buffer, 64, "test command: <AT%s=?> is executed\r\
    ↵n", cmd_name);
    esp_at_port_write_data(buffer, strlen((char *)buffer));

    return ESP_AT_RESULT_CODE_OK;
}
```

- 查询命令：

```
static uint8_t at_query_cmd_test(uint8_t *cmd_name)
{
    uint8_t buffer[64] = {0};
    snprintf((char *)buffer, 64, "query command: <AT%s?> is executed\r\
    ↵n", cmd_name);
    esp_at_port_write_data(buffer, strlen((char *)buffer));

    return ESP_AT_RESULT_CODE_OK;
}
```

- 设置命令：

```
static uint8_t at_setup_cmd_test(uint8_t para_num)
{
    uint8_t index = 0;

    // 获取第一个参数，并将其解析为一个数字
    int32_t digit = 0;
    if (esp_at_get_para_as_digit(index++, &digit) != ESP_AT_PARA_PARSE_
    ↵RET_OK) {
        return ESP_AT_RESULT_CODE_ERROR;
    }

    // 获取第二个参数，并将其解析为一个字符串
    uint8_t *str = NULL;
    if (esp_at_get_para_as_str(index++, &str) != ESP_AT_PARA_PARSE_RET_
    ↵OK) {
        return ESP_AT_RESULT_CODE_ERROR;
    }

    // 分配一个缓冲区，构建数据，然后通过接口 (uart/spi/sdio/socket) ↵
    ↵将数据发送到 MCU
    uint8_t *buffer = (uint8_t *)malloc(512);
    if (!buffer) {
        return ESP_AT_RESULT_CODE_ERROR;
    }
    int len = snprintf((char *)buffer, 512, "setup command: <AT%s=%d,\
    ↵%s\> is executed\r\n",
```

(续下页)

(接上页)

```

        esp_at_get_current_cmd_name(), digit, str);
    esp_at_port_write_data(buffer, len);

    // 记得释放缓冲区
    free(buffer);

    return ESP_AT_RESULT_CODE_OK;
}

```

- 执行命令：

```

static uint8_t at_exe_cmd_test(uint8_t *cmd_name)
{
    uint8_t buffer[64] = {0};
    snprintf((char *)buffer, 64, "execute command: <AT%s> is executed\
↪r\n", cmd_name);
    esp_at_port_write_data(buffer, strlen((char *)buffer));

    return ESP_AT_RESULT_CODE_OK;
}

```

第二步：定义注册 AT 命令函数

- 请定义 `esp_at_custom_cmd_register` 函数，并在函数中调用 API `esp_at_custom_cmd_array_register()` 注册 AT 命令。
示例代码：

```

bool esp_at_custom_cmd_register(void)
{
    return esp_at_custom_cmd_array_register(at_custom_cmd, sizeof(at_custom_
↪cmd) / sizeof(esp_at_cmd_t));
}

```

- 然后，调用 API `ESP_AT_CMD_SET_INIT_FN` 来初始化你实现的注册 AT 命令函数 `esp_at_custom_cmd_register`。
示例代码：

```
ESP_AT_CMD_SET_INIT_FN(esp_at_custom_cmd_register, 1);
```

备注： 如果你是在 `examples/at_custom_cmd/custom` 和 `examples/at_custom_cmd/include` 目录下创建新的源文件和头文件自定义 AT 命令，请避免将注册函数命名为 `esp_at_custom_cmd_register`，因为该函数在 `at_custom_cmd` 示例中已被定义和初始化。你可以将函数命名为 `esp_at_custom_cmd_register_foo`，然后使用 `ESP_AT_CMD_SET_INIT_FN` 初始化该函数。

第三步：增加组件依赖

如果你在 **第一步：定义 AT 命令** 时使用了除 `at`、`freertos`、`nvs_flash` 外其他组件，请在 `examples/at_custom_cmd/CMakeLists.txt` 文件中添加这些组件依赖。反之，可以跳过此步骤。例如新增使用 `lwip` 组件，则示例代码如下：

```
set(require_components at freertos nvs_flash lwip)
```

第四步：增加链接选项

请在 `examples/at_custom_cmd/CMakeLists.txt` 文件中，将自定义的注册 AT 命令函数名称作为一个链接选项强制链接给 `${COMPONENT_LIB}`，确保程序运行时可以找到该函数。示例代码如下：

```
target_link_libraries(${COMPONENT_LIB} INTERFACE "-u esp_at_custom_cmd_register")
```

备注：如果自定义的注册 AT 命令函数名称为 `esp_at_custom_cmd_register_foo`，则示例代码如下：

```
target_link_libraries(${COMPONENT_LIB} INTERFACE "-u esp_at_custom_cmd_register_foo
↪")
```

第五步：设置组件环境变量

本节介绍了两种设置 `at_custom_cmd` 组件环境变量的方法，以确保 ESP-AT 项目在编译时能够正确找到该组件。根据你的需求选择适合的方法。如果你在 `esp-at/components` 目录下的原始组件中自定义 AT 命令或修改代码，则无需执行此步骤。但不建议在 `esp-at/components` 目录下的原始组件中自定义 AT 命令，本文也不对此进行说明。

方法 1：在命令行中设置 `AT_CUSTOM_COMPONENTS` 环境变量（适用于本地编译）。

- Linux or macOS

```
export AT_CUSTOM_COMPONENTS=(path_of_at_custom_cmd)
```

- Windows

```
set AT_CUSTOM_COMPONENTS=(path_of_at_custom_cmd)
```

备注：

- 请将 `(path_of_at_custom_cmd)` 替换为 `at_custom_cmd` 目录的真实绝对路径。
- 你可以指定多个组件。例如：

```
export AT_CUSTOM_COMPONENTS="~/prefix/my_path1 ~/prefix/
my_path2"
```

方法 2：在 `esp-at/build.py` 文件 `setup_env_variables()` 函数中加入设置 `AT_CUSTOM_COMPONENTS` 环境变量的代码（适用于本地编译和网页编译）。示例代码如下：

```
# set AT_CUSTOM_COMPONENTS
at_custom_cmd_path=os.path.join(os.getcwd(), 'examples/at_custom_cmd')
os.environ['AT_CUSTOM_COMPONENTS']=at_custom_cmd_path
```

第六步：编译 AT 固件

完成以上步骤后，可根据需要选择通过网页编译或本地编译 AT 固件，并将其烧录到你的设备上。

6.4.2 运行 AT+TEST 命令获取运行结果

正确操作上面步骤后，运行 `AT+TEST` 命令获取结果如下。

测试命令：


```
AT+TEST=?
```

响应:

```
AT+TEST=?
test command: <AT+TEST=?> is executed

OK
```

查询命令:

```
AT+TEST?
```

响应:

```
AT+TEST?
query command: <AT+TEST?> is executed

OK
```

设置命令:

```
AT+TEST=1,"espressif"
```

响应:

```
AT+TEST=1,"espressif"
setup command: <AT+TEST=1,"espressif"> is executed

OK
```

执行命令:

```
AT+TEST
```

响应:

```
AT+TEST
execute command: <AT+TEST> is executed

OK
```

6.4.3 自定义复杂的 AT 命令

下面列举的示例代码适用于定义更加复杂的命令，请根据实际需要进行自定义。

定义返回消息

ESP-AT 已经在 `esp_at_rc_t` 定义了一些返回消息，更多返回消息请参见 [AT 消息](#)。

除了通过 `return` 模式返回消息，也可调用 API `esp_at_write_result()` 来返回命令执行结果。可在代码中同时使用 API 和 `ESP_AT_RESULT_CODE_SEND_OK` 及 `ESP_AT_RESULT_CODE_SEND_FAIL`。

例如，当使用 AT+TEST 的执行命令向服务器或 MCU 发送数据时，用 `esp_at_write_result()` 来返回发送结果，用 `return` 模式来返回命令执行结果，示例代码如下。

```
uint8_t at_exe_cmd_test(uint8_t *cmd_name)
{
    uint8_t buffer[64] = {0};
```

(续下页)

(接上页)

```

snprintf((char *)buffer, 64, "this cmd is execute cmd: %s\r\n", cmd_name);

esp_at_port_write_data(buffer, strlen((char *)buffer));

// 向服务器或 MCU 发送数据的自定义操作
send_data_to_server();

// 返回 SEND_OK
esp_at_write_result(ESP_AT_RESULT_CODE_SEND_OK);

return ESP_AT_RESULT_CODE_OK;
}

```

运行命令及返回的响应：

```

AT+TEST
this cmd is execute cmd: +TEST

SEND OK

OK

```

获取命令参数

ESP-AT 提供以下两个 API 获取命令参数。

- `esp_at_get_para_as_digit()` 可获取数字参数。
- `esp_at_get_para_as_str()` 可获取字符串参数。

示例请见设置命令。

省略命令参数

本节介绍如何设置某些命令参数为可选参数。

- 省略首位或中间参数
- 省略最后一位参数

省略首位或中间参数 假设你想将 AT+TEST 命令的 <param_2> 和 <param_3> 参数设置为可选参数，其中 <param_2> 为数字参数，<param_3> 为字符串参数。

```
AT+TEST=<param_1>[,<param_2>][,<param_3>],<param_4>
```

实现代码如下。

```

uint8_t at_setup_cmd_test(uint8_t para_num)
{
    int32_t para_int_1 = 0;
    int32_t para_int_2 = 0;
    uint8_t *para_str_3 = NULL;
    uint8_t *para_str_4 = NULL;
    uint8_t num_index = 0;
    uint8_t buffer[64] = {0};
    esp_at_para_parse_ret_t parse_result = ESP_AT_PARA_PARSE_RET_OK;

    snprintf((char *)buffer, 64, "this cmd is setup cmd and cmd num is: %u\r\n",
    ↪ para_num);

```

(续下页)

(接上页)

```

esp_at_port_write_data(buffer, strlen((char *)buffer));

parse_result = esp_at_get_para_as_digit(num_index++, &para_int_1);
if (parse_result != ESP_AT_PARA_PARSE_RET_OK) {
    return ESP_AT_RESULT_CODE_ERROR;
} else {
    memset(buffer, 0, 64);
    snprintf((char *)buffer, 64, "first parameter is: %d\r\n", para_int_1);
    esp_at_port_write_data(buffer, strlen((char *)buffer));
}

parse_result = esp_at_get_para_as_digit(num_index++, &para_int_2);
if (parse_result != ESP_AT_PARA_PARSE_RET_OMITTED) {
    if (parse_result != ESP_AT_PARA_PARSE_RET_OK) {
        return ESP_AT_RESULT_CODE_ERROR;
    } else {
        // 示例代码
        // 需要自定义操作
        memset(buffer, 0, 64);
        snprintf((char *)buffer, 64, "second parameter is: %d\r\n", para_int_
↪2);
        esp_at_port_write_data(buffer, strlen((char *)buffer));
    }
} else {
    // 示例代码
    // 省略第二个参数
    // 需要自定义操作
    memset(buffer, 0, 64);
    snprintf((char *)buffer, 64, "second parameter is omitted\r\n");
    esp_at_port_write_data(buffer, strlen((char *)buffer));
}

parse_result = esp_at_get_para_as_str(num_index++, &para_str_3);
if (parse_result != ESP_AT_PARA_PARSE_RET_OMITTED) {
    if (parse_result != ESP_AT_PARA_PARSE_RET_OK) {
        return ESP_AT_RESULT_CODE_ERROR;
    } else {
        // 示例代码
        // 需自定义操作
        memset(buffer, 0, 64);
        snprintf((char *)buffer, 64, "third parameter is: %s\r\n", para_str_3);
        esp_at_port_write_data(buffer, strlen((char *)buffer));
    }
} else {
    // 示例代码
    // 省略第三个参数
    // 需自定义操作
    memset(buffer, 0, 64);
    snprintf((char *)buffer, 64, "third parameter is omitted\r\n");
    esp_at_port_write_data(buffer, strlen((char *)buffer));
}

parse_result = esp_at_get_para_as_str(num_index++, &para_str_4);
if (parse_result != ESP_AT_PARA_PARSE_RET_OK) {
    return ESP_AT_RESULT_CODE_ERROR;
} else {
    memset(buffer, 0, 64);
    snprintf((char *)buffer, 64, "fourth parameter is: %s\r\n", para_str_4);
    esp_at_port_write_data(buffer, strlen((char *)buffer));
}

```

(续下页)

(接上页)

```

return ESP_AT_RESULT_CODE_OK;
}

```

备注：如果输入的字符串参数为 "", 则该参数没有被省略。

省略最后一位参数 假设 AT+TEST 命令的 <param_3> 参数为字符串参数, 且为最后一位参数, 你想将它设置为可选参数。

```
AT+TEST=<param_1>,<param_2>[,<param_3>]
```

则有以下两种省略情况。

- AT+TEST=<param_1>,<param_2>
- AT+TEST=<param_1>,<param_2>,

实现代码如下。

```

uint8_t at_setup_cmd_test(uint8_t para_num)
{
    int32_t para_int_1 = 0;
    uint8_t *para_str_2 = NULL;
    uint8_t *para_str_3 = NULL;
    uint8_t num_index = 0;
    uint8_t buffer[64] = {0};
    esp_at_para_parse_ret_t parse_result = ESP_AT_PARA_PARSE_RET_OK;

    snprintf((char *)buffer, 64, "this cmd is setup cmd and cmd num is: %u\r\n",
    ↪ para_num);
    esp_at_port_write_data(buffer, strlen((char *)buffer));

    parse_result = esp_at_get_para_as_digit(num_index++, &para_int_1);
    if (parse_result != ESP_AT_PARA_PARSE_RET_OK) {
        return ESP_AT_RESULT_CODE_ERROR;
    } else {
        memset(buffer, 0, 64);
        snprintf((char *)buffer, 64, "first parameter is: %d\r\n", para_int_1);
        esp_at_port_write_data(buffer, strlen((char *)buffer));
    }

    parse_result = esp_at_get_para_as_str(num_index++, &para_str_2);
    if (parse_result != ESP_AT_PARA_PARSE_RET_OK) {
        return ESP_AT_RESULT_CODE_ERROR;
    } else {
        memset(buffer, 0, 64);
        snprintf((char *)buffer, 64, "second parameter is: %s\r\n", para_str_2);
        esp_at_port_write_data(buffer, strlen((char *)buffer));
    }

    if (num_index == para_num) {
        memset(buffer, 0, 64);
        snprintf((char *)buffer, 64, "third parameter is omitted\r\n");
        esp_at_port_write_data(buffer, strlen((char *)buffer));
    } else {
        parse_result = esp_at_get_para_as_str(num_index++, &para_str_3);
        if (parse_result != ESP_AT_PARA_PARSE_RET_OMITTED) {
            if (parse_result != ESP_AT_PARA_PARSE_RET_OK) {
                return ESP_AT_RESULT_CODE_ERROR;
            } else {

```

(续下页)

(接上页)

```

        // 示例代码
        // 需自定义操作
        memset(buffer, 0, 64);
        snprintf((char *)buffer, 64, "third parameter is: %s\r\n", para_
↪str_3);

        esp_at_port_write_data(buffer, strlen((char *)buffer));
    }
} else {
    // 示例代码
    // 省略第三个参数
    // 需自定义操作
    memset(buffer, 0, 64);
    snprintf((char *)buffer, 64, "third parameter is omitted\r\n");
    esp_at_port_write_data(buffer, strlen((char *)buffer));
}
}

return ESP_AT_RESULT_CODE_OK;
}

```

备注：如果输入的字符串参数为 "", 则该参数没有被省略。

阻塞命令的执行

有时你想阻塞某个命令的执行，等待另一个执行结果，然后系统基于这个结果可能会返回不同的值。

一般来说，这类命令需要与其它任务的结果进行同步。

推荐使用 semaphore 来同步。

示例代码如下。

```

xSemaphoreHandle at_operation_sema = NULL;

uint8_t at_exe_cmd_test(uint8_t *cmd_name)
{
    uint8_t buffer[64] = {0};

    snprintf((char *)buffer, 64, "this cmd is execute cmd: %s\r\n", cmd_name);

    esp_at_port_write_data(buffer, strlen((char *)buffer));

    // 示例代码
    // 不必在此处创建 semaphores
    at_operation_sema = xSemaphoreCreateBinary();
    assert(at_operation_sema != NULL);

    // 阻塞命令的执行
    // 等待另一个执行的结果
    // 其它任务可调用 xSemaphoreGive 来释放 semaphore
    xSemaphoreTake(at_operation_sema, portMAX_DELAY);

    return ESP_AT_RESULT_CODE_OK;
}

```

从 AT 命令端口获取输入的数据

ESP-AT 支持从 AT 命令端口访问输入的数据，为此提供以下两个 API。

- `esp_at_port_enter_specific()` 设置回调函数，AT 端口接收到输入的数据后，将调用该函数。
- `esp_at_port_exit_specific()` 删除由 `esp_at_port_enter_specific` 设置的回调函数。

获取数据的方法会根据数据长度是否被指定而有所不同。

指定长度的输入数据 假设你已经使用 `<param_1>` 指定了数据长度，如下所示。

```
AT+TEST=<param_1>
```

以下示例代码介绍如何从 AT 命令端口获取长度为 `<param_1>` 的输入数据。

```
static xSemaphoreHandle at_sync_sema = NULL;

void wait_data_callback(void)
{
    xSemaphoreGive(at_sync_sema);
}

uint8_t at_setup_cmd_test(uint8_t para_num)
{
    int32_t specified_len = 0;
    int32_t received_len = 0;
    int32_t remain_len = 0;
    uint8_t *buf = NULL;
    uint8_t buffer[64] = {0};

    if (esp_at_get_para_as_digit(0, &specified_len) != ESP_AT_PARA_PARSE_RET_OK) {
        return ESP_AT_RESULT_CODE_ERROR;
    }

    buf = (uint8_t *)malloc(specified_len);
    if (buf == NULL) {
        memset(buffer, 0, 64);
        snprintf((char *)buffer, 64, "malloc failed\r\n");
        esp_at_port_write_data(buffer, strlen((char *)buffer));
    }

    // 示例代码
    // 不必在此处创建 semaphores
    if (!at_sync_sema) {
        at_sync_sema = xSemaphoreCreateBinary();
        assert(at_sync_sema != NULL);
    }

    // 返回输入数据提示符 ">"
    esp_at_port_write_data((uint8_t *)>, strlen(">"));

    // 设置回调函数，在接收到输入数据后由 AT 端口调用
    esp_at_port_enter_specific(wait_data_callback);

    // 接收输入的数据
    while(xSemaphoreTake(at_sync_sema, portMAX_DELAY)) {
        received_len += esp_at_port_read_data(buf + received_len, specified_len -
        ↪received_len);

        if (specified_len == received_len) {
            esp_at_port_exit_specific();

            // 获取剩余输入数据的长度
            remain_len = esp_at_port_get_data_length();
            if (remain_len > 0) {
```

(续下页)

(接上页)

```

        // 示例代码
        // 如果剩余数据长度 > 0, 则实际输入数据长度大于指定的接收数据长度
        // 可自定义如何处理这些剩余数据
        // 此处只是简单打印出剩余数据
        esp_at_port_recv_data_notify(remain_len, portMAX_DELAY);
    }

    // 示例代码
    // 输出接收到的数据
    memset(buffer, 0, 64);
    snprintf((char *)buffer, 64, "\r\nreceived data is: ");
    esp_at_port_write_data(buffer, strlen((char *)buffer));

    esp_at_port_write_data(buf, specified_len);

    break;
}

}

free(buf);

return ESP_AT_RESULT_CODE_OK;
}

```

因此, 如果你设置 AT+TEST=5, 输入的数据为 1234567890, 那么 ESP-AT 返回的结果如下所示。

```

AT+TEST=5
>67890
received data is: 12345
OK

```

未指定长度的输入数据 这种情况类似 Network 透传模式, 不指定数据长度。

```
AT+TEST
```

假设 ESP-AT 结束命令的执行并返回执行结果, 示例代码如下。

```

#define BUFFER_LEN (2048)
static xSemaphoreHandle at_sync_sema = NULL;

void wait_data_callback(void)
{
    xSemaphoreGive(at_sync_sema);
}

uint8_t at_exe_cmd_test(uint8_t *cmd_name)
{
    int32_t received_len = 0;
    int32_t remain_len = 0;
    uint8_t *buf = NULL;
    uint8_t buffer[64] = {0};

    buf = (uint8_t *)malloc(BUFFER_LEN);
    if (buf == NULL) {
        memset(buffer, 0, 64);
        snprintf((char *)buffer, 64, "malloc failed\r\n");
        esp_at_port_write_data(buffer, strlen((char *)buffer));
    }
}

```

(续下页)

```

// 示例代码
// 不必在此处创建 semaphores
if (!at_sync_sema) {
    at_sync_sema = xSemaphoreCreateBinary();
    assert(at_sync_sema != NULL);
}

// 返回输入数据提示符 ">"
esp_at_port_write_data((uint8_t *)>", strlen(">"));

// 设置回调函数，在接收到输入数据后由 AT 端口调用
esp_at_port_enter_specific(wait_data_callback);

// 接收输入的数据
while(xSemaphoreTake(at_sync_sema, portMAX_DELAY)) {
    memset(buf, 0, BUFFER_LEN);

    received_len = esp_at_port_read_data(buf, BUFFER_LEN);
    // 检查是否退出该模式
    // 退出条件是接收到 "+++" 字符串
    if ((received_len == 3) && (strncmp((const char *)buf, "+++", 3)) == 0) {
        esp_at_port_exit_specific();

        // 示例代码
        // 如果剩余数据长度 > 0，说明缓冲区内仍有数据需要处理
        // 可自定义如何处理剩余数据
        // 此处只是简单打印出剩余数据
        remain_len = esp_at_port_get_data_length();
        if (remain_len > 0) {
            esp_at_port_recv_data_notify(remain_len, portMAX_DELAY);
        }

        break;
    } else if (received_len > 0) {
        // 示例代码
        // 可自定义如何处理接收到的数据
        // 此处只是简单打印出接收到的数据
        memset(buffer, 0, 64);
        snprintf((char *)buffer, 64, "\r\nreceived data is: ");
        esp_at_port_write_data(buffer, strlen((char *)buffer));

        esp_at_port_write_data(buf, strlen((char *)buf));
    }
}

free(buf);

return ESP_AT_RESULT_CODE_OK;
}

```

因此，如果第一个输入数据是 1234567890，第二个输入数据是 +++，那么 ESP-AT 返回结果如下所示。

```

AT+TEST
>
received data is: 1234567890
OK

```


6.5 如何提高 ESP-AT 吞吐性能

默认情况下，ESP-AT 和主机之间使用 UART 进行通信，因此最高吞吐速率不会超过其默认配置，即不会超过 115200 bps。用户如要 ESP-AT 实现较高的吞吐量，需了解本文，并做出有针对性的配置。本文以 ESP32-C6 为例，介绍如何提高 ESP-AT 吞吐性能。

备注：本文基于 ESP-AT 处于透传模式下，描述提高 TCP/UDP/SSL 吞吐性能的一般性方法。

用户可以选择下面其中一种方法，提高吞吐性能：

- (简单) 快速配置
- (推荐) 熟悉数据流、针对性地配置

6.5.1 (简单) 快速配置

1. 配置系统、LWIP、Wi-Fi 适用于高吞吐的参数

将下面代码段拷贝并追加至 `module_config/module_esp32c6_default/sdkconfig.defaults` 文件最后，其它 ESP32-C6 系列设备请修改对应文件夹下的 `sdkconfig.defaults` 文件。

```
# System
CONFIG_ESP_SYSTEM_EVENT_TASK_STACK_SIZE=4096
CONFIG_FREERTOS_UNICORE=n
CONFIG_FREERTOS_HZ=1000
CONFIG_ESP_DEFAULT_CPU_FREQ_160=y
CONFIG_ESP_DEFAULT_CPU_FREQ_MHZ=160
CONFIG_ESPTOOLPY_FLASHMODE_QIO=y
CONFIG_ESPTOOLPY_FLASHFREQ_80M=y

# LWIP
CONFIG_LWIP_TCP_SND_BUF_DEFAULT=65534
CONFIG_LWIP_TCP_WND_DEFAULT=65534
CONFIG_LWIP_TCP_RECVMBOX_SIZE=12
CONFIG_LWIP_UDP_RECVMBOX_SIZE=12
CONFIG_LWIP_TCPIP_RECVMBOX_SIZE=64

# Wi-Fi
CONFIG_ESP_WIFI_STATIC_RX_BUFFER_NUM=16
CONFIG_ESP_WIFI_DYNAMIC_RX_BUFFER_NUM=64
CONFIG_ESP_WIFI_DYNAMIC_TX_BUFFER_NUM=64
CONFIG_ESP_WIFI_TX_BA_WIN=32
CONFIG_ESP_WIFI_RX_BA_WIN=32
CONFIG_ESP_WIFI_AMPDU_TX_ENABLED=y
CONFIG_ESP_WIFI_AMPDU_RX_ENABLED=y
```

2. 提高 UART 缓冲区大小

将下面代码段拷贝并替换 `at_uart_task.c` 文件中 `uart_driver_install()` 行。

```
uart_driver_install(g_at_cmd_port, 1024 * 16, 1024 * 16, 100, &s_at_uart_
↪queue, 0);
```

3. 删除默认配置、重新编译固件、烧录运行

```
rm -rf build sdkconfig
./build.py build
./build.py flash monitor
```

4. 透传前提高 UART 波特率

典型 AT 命令序列如下：

```
AT+CWMODE=1
AT+CWJAP="ssid","password"
AT+UART_CUR=3000000,8,1,0,3
AT+CIPSTART="TCP","192.168.105.13",3344
AT+CIPMODE=1
AT+CIPSEND
// 传输数据
```

此快速配置的方法在一定程度上可以提高吞吐，但有时可能不能达到用户的预期。另外有些配置可能不是吞吐的瓶颈，配置较高可能会牺牲内存资源或功耗等。因此，用户也可以熟悉下面推荐的方法，进行有针对性地配置。

6.5.2 （推荐）熟悉数据流、针对性地配置

影响 ESP-AT 吞吐的因素大体描述如下图：

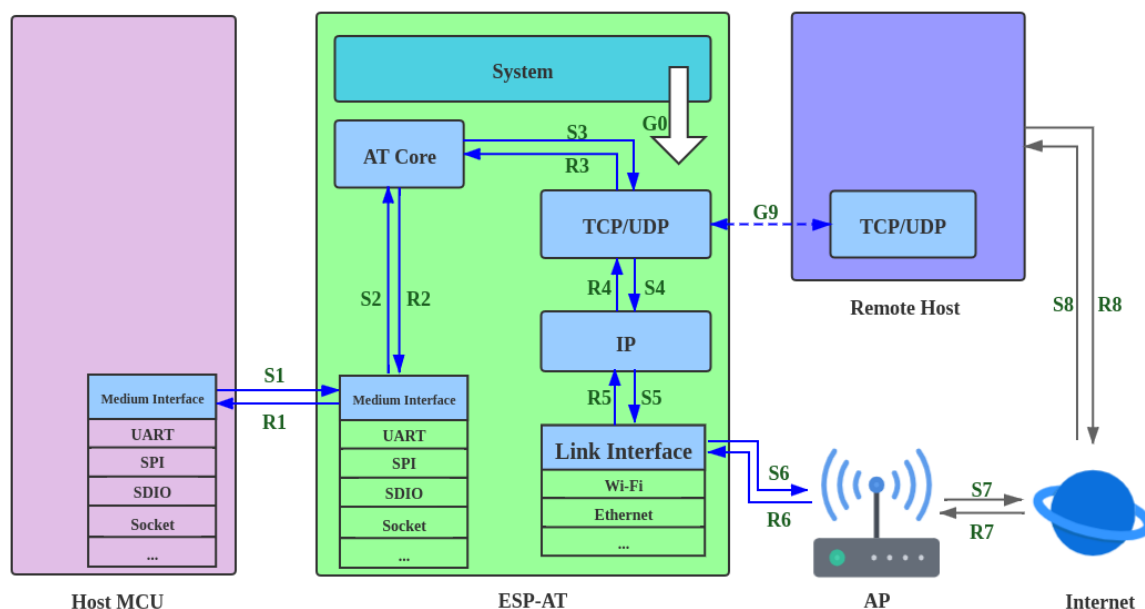


图 20: 吞吐数据流

如图中箭头所示：

- ESP-AT 发送 (TX) 的数据流为 S1 -> S2 -> S3 -> S4 -> S5 -> S6 -> S7 -> S8
- ESP-AT 接收 (RX) 的数据流为 R8 -> R7 -> R6 -> R5 -> R4 -> R3 -> R2 -> R1

吞吐的数据流类似于水流，要想提高吞吐，需要考虑在数据流速较低的节点之间进行优化，而不需要在数据流速本已达到预期的节点之间进行额外的配置，以免造成不必要的资源浪费。在实际产品中，往往只需要提高其中一条数据流吞吐即可，用户需要根据下面指导进行对应的配置即可。

备注： 下面的配置均以可用内存充足为前提的，用户可以通过 `AT+SYSRAM` 命令来查询可用内存。

1. G0 吞吐优化

G0 是系统可以优化的部分，建议参考配置如下：

```
CONFIG_ESP_SYSTEM_EVENT_TASK_STACK_SIZE=4096
CONFIG_FREERTOS_UNICORE=n
CONFIG_FREERTOS_HZ=1000
CONFIG_ESP_DEFAULT_CPU_FREQ_160=y
CONFIG_ESP_DEFAULT_CPU_FREQ_MHZ=160
CONFIG_ESPTOOLPY_FLASHMODE_QIO=y
CONFIG_ESPTOOLPY_FLASHFREQ_80M=y
```

2. S1、R1 吞吐优化

通常情况下，S1 和 R1 是 ESP-AT 吞吐高低的关键。因为默认情况下，ESP-AT 和主机之间使用 UART 进行通信，波特率为 115200，而 UART 硬件上，速率上限为 5 Mbps。因此，用户使用场景吞吐低于 5 Mbps，可以使用默认的 UART 作为和主机之间的通信介质，同时可以进行下面优化。

2.1 提高 UART 缓冲区大小

将下面代码段拷贝并替换 `at_uart_task.c` 文件中 `uart_driver_install()` 行。

- 提高 UART TX 吞吐

```
uart_driver_install(g_at_cmd_port, 1024 * 16, 8192, 100, &s_at_uart_
↪queue, 0);
```

- 提高 UART RX 吞吐

```
uart_driver_install(g_at_cmd_port, 2048, 1024 * 16, 100, &s_at_uart_
↪queue, 0);
```

- 提高 UART TX 和 RX 吞吐

```
uart_driver_install(g_at_cmd_port, 1024 * 16, 1024 * 16, 100, &s_at_
↪uart_queue, 0);
```

2.2 透传前提高 UART 波特率

典型 AT 命令序列如下：

```
AT+CWMODE=1
AT+CWJAP="ssid","password"
AT+UART_CUR=3000000,8,1,0,3
AT+CIPSTART="TCP","192.168.105.13",3344
AT+CIPMODE=1
AT+CIPSEND
// 传输数据
```

备注： 用户需要确保主机的 UART 可以支持到这么高的速率，并且主机和 ESP-AT 之间的 UART 连线尽可能地短。

备注： 如果用户期望吞吐速率大于或接近于 5 Mbps，可以考虑使用 SPI、SDIO、Socket 等方式。具体请参考：

- SDIO: [SDIO AT 指南](#)
- SPI: [SPI AT 指南](#)
- Socket: [Socket AT 指南](#)

3. S2、R2、R3、S3 吞吐优化

通常情况下，S2、R2、R3、S3 不是 ESP-AT 吞吐高低的瓶颈。因为 AT core 在 UART 缓冲区和通信协议的传输层之间传递数据，仅有极少的且不耗时的应用逻辑，无需优化。

4. S4、R4、S5、R5、S6、R6 吞吐优化

ESP-AT 和主机之间使用 UART 进行通信，S4、R4、S5、R5、S6、R6 无需优化。ESP-AT 和主机之间使用其他传输介质进行通信时，S4、R4、S5、R5、S6、R6 是影响吞吐的一个因素。

S4、R4、S5、R5、S6、R6 是通信协议的传输层、网络层、和数据链路层之间的数据流。用户需要阅读 ESP-IDF 中 [如何提高 Wi-Fi 性能](#) 文档，了解原理，进行合理配置。这些配置均可以在 `./build.py menuconfig` 里进行配置。

- 优化 S4 -> S5 -> S6: [发送数据方向配置](#)
- 优化 R6 -> R5 -> R4: [接收数据方向配置](#)

5. S6、R6 吞吐优化

S6 和 R6 是通信协议的数据链路层，ESP32-C6 可以使用 Wi-Fi 或者以太网作为传输介质。Wi-Fi 除了上述介绍的优化方法之外，可能还需要用户关心：

- 提高 RF 发射功率
默认发射功率通常不是吞吐高低的瓶颈，用户也可以通过 [AT+RFPOWER](#) 命令查询和设置 RF 发射功率。
- 设置 802.11 b/g/n 协议
默认 Wi-Fi 模式即为 802.11 b/g/n 协议，用户可通过 [AT+CWSTAPROTO](#) 命令查询和设置 802.11 b/g/n 协议。配置是双向的，因此建议 AP 端 Wi-Fi 模式配置为 802.11 b/g/n 协议，频宽配置为 HT20/HT40 (20/40 MHz) 模式。

6. S7、R7、S8、R8 吞吐优化

通常情况下，S7、R7、S8、R8 不是 ESP-AT 吞吐优化的范围。因为这和实际网络带宽、网络路由、物理距离等有关。

6.6 如何更新 mfg_nvs 分区

6.6.1 mfg_nvs 分区介绍

`mfg_nvs` (*manufacturing nvs*) 分区定义在 `esp-at/module_config/{module_name}/at_customize.csv` 文件中。该分区存储了出厂固件的下列配置：

- 出厂参数配置 (Wi-Fi 配置、UART 配置、模组配置)：请参考[出厂参数配置介绍](#)。
- PKI 配置 (各类证书和密钥配置)：请参考[PKI 配置介绍](#)。
- GATTS 配置 (低功耗蓝牙服务)：请参考[低功耗蓝牙服务源文件介绍](#)。

当你需要修改出厂参数配置、PKI 配置或者 GATTS 配置时，你可以重新编译 ESP-AT 工程，生成新的 `mfg_nvs.bin` 文件；或者通过 [at.py 工具](#) 修改固件，为你的模组生成新的固件。本文介绍前者。

6.6.2 生成 mfg_nvs.bin

配置修改完成后，重新编译 [ESP-AT 工程](#)。在编译阶段会自动通过 `mfg_nvs.py` 先生成 `build/customized_partitions/mfg_nvs.csv` 文件 (包含了所有配置的命名空间、键、类型、和键值信息)，再由 `esp-idf` 中的 `nvs_partition_gen.py` 脚本生成 `build/customized_partitions/mfg_nvs.bin` 文件 ([NVS 库的结构](#))。

6.6.3 下载 mfg_nvs.bin

可采用以下任意一种方式下载 `mfg_nvs.bin` 文件。

- 下载重新编译过的 ESP-AT 固件，详情请见[第七步：烧录到设备](#)。

- 仅下载 `mfg_nvs.bin`，这种方法只更新 ESP32-C6 中的 `mfg_nvs` 分区。
 - Windows

请下载 [Windows Flash 下载工具](#)。详细指导可参阅 [Flash 下载工具用户指南](#)。下载 `build/customized_partitions/mfg_nvs.bin` 文件到 ESP32-C6。你可以通过 [AT+SYSFLASH?](#) 命令查询 `mfg_nvs.bin` 的下载地址。
 - Linux or macOS

请使用 [esptool.py](#)。

你可以在 ESP-AT 根目录执行以下命令下载 `mfg_nvs.bin` 文件。

```
esptool.py --chip auto --port PORTNAME --baud 921600 --before_
↪default_reset --after hard_reset write_flash -z --flash_mode dio -
↪-flash_freq 40m --flash_size 4MB ADDRESS mfg_nvs.bin
```

将 `PORTNAME` 替换为你的串口名称。ADDRESS 替换为下载 `mfg_nvs.bin` 文件的地址，你可以通过 [AT+SYSFLASH?](#) 命令查询下载地址。

- MCU 发送 [AT+SYSFLASH](#) 命令更新 ESP32-C6 中的 `mfg_nvs` 分区。

```
# 擦除 mfg_nvs 分区
AT+SYSFLASH=0, "mfg_nvs", 0, MFG_NVS_SIZE

# 写入 mfg_nvs.bin 文件
AT+SYSFLASH=1, "mfg_nvs", 0, MFG_NVS_SIZE
```

`MFG_NVS_SIZE` 替换为下载 `mfg_nvs.bin` 文件的大小，不同的模组有不同的分区大小，你可以通过 [AT+SYSFLASH?](#) 命令查询分区大小。

6.7 如何更新出厂参数

本文档介绍了如何通过重新编译固件来更新 ESP-AT 默认的出厂参数配置。出厂参数配置包括一些 Wi-Fi 配置、UART 配置、模组配置。

备注：如果你想在运行时通过 AT 命令更新出厂参数而无需重新编译固件，请参考 [AT+SYSMFG](#) 命令示例。

- [出厂参数配置介绍](#)
- [生成 `mfg\_nvs.bin` 文件](#)
- [下载 `mfg\_nvs.bin` 文件](#)

6.7.1 出厂参数配置介绍

当前默认的出厂参数配置的源文件 `customized_partitions/raw_data/factory_param/factory_param_data.csv`，如下所示：

功能	当前配置	相关 AT 命令
版本信息	version (出厂参数结构的版本号, 当前版本为 4, 用于标识参数格式和兼容性)	AT+SYSMFG
Wi-Fi 配置	- max_tx_power (ESP32-C6 的 Wi-Fi 最大发射功率, 详情请见 ESP32-C6 发射功率 设置范围) - country_code (国家代码) - start_channel (起始 Wi-Fi 信道) - channel_num (Wi-Fi 的总信道数量)	所有需要 Wi-Fi 功能的 AT 命令
UART 配置	- uart_port (用于发送 AT 命令和接收 AT 响应的 UART 端口) - uart_baudrate (UART 波特率) - uart_tx_pin (UART TX 引脚) - uart_rx_pin (UART RX 引脚) - uart_cts_pin (UART CTS 引脚) - uart_rts_pin (UART RTS 引脚)	所有需要 UART 功能的 AT 命令
系统存储模式	sys_store (系统默认存储模式) 0: 不自动存储配置更改到 flash 1: 自动存储配置更改到 flash (默认值)	AT+SYSSTORE
模组名称	module_name	AT+GMR

请根据自己的需求修改出厂参数配置, 然后生成 `mfg_nvs.bin` 文件。

6.7.2 生成 `mfg_nvs.bin` 文件

请参考 [生成 `mfg\_nvs.bin`](#) 文档生成带有出厂参数配置的 `mfg_nvs.bin`。

6.7.3 下载 `mfg_nvs.bin` 文件

请参考 [下载 `mfg\_nvs.bin`](#) 文档。

6.8 如何更新 PKI 配置

本文档介绍了如何通过重新编译固件来更新 ESP-AT 提供的默认的 [PKI](#) 配置。PKI 配置包括 TLS 客户端、TLS 服务器、MQTT 客户端、WebSocket 客户端、HTTP 客户端和 Wi-Fi Enterprise 客户端的证书和密钥。

备注: 如果你想在运行时通过 AT 命令更新证书而无需重新编译固件, 请参考 [AT+SYSMFG 命令示例](#)。

- [PKI 配置介绍](#)
- [生成 `mfg\_nvs.bin` 文件](#)
- [下载 `mfg\_nvs.bin` 文件](#)

6.8.1 PKI 配置介绍

当前默认 PKI 配置的源文件位于 `customized_partitions/raw_data` 目录下，如下所示：

功能	当前配置	相关 AT 命令
TLS 客户端	第 0 套客户端配置 • <code>client_ca_00.crt</code> • <code>client_cert_00.crt</code> • <code>client_key_00.key</code> 第 1 套客户端配置 • <code>client_ca_01.crt</code> • <code>client_cert_01.crt</code> • <code>client_key_01.key</code>	• <i><code>AT+CIPSTART</code></i> • <i><code>AT+CIPSSLCCONF</code></i>
TLS 服务器	• <code>server_ca.crt</code> • <code>server_cert.crt</code> • <code>server.key</code>	<i><code>AT+CIPSERVER</code></i>
MQTT 客户端	• <code>mqtt_ca.crt</code> • <code>mqtt_client.crt</code> • <code>mqtt_client.key</code>	<i><code>AT+MQTTUSERCFG</code></i>
WebSocket 客户端	第 0 套客户端配置 • <code>wss_ca_00.crt</code> • <code>wss_client_00.crt</code> • <code>wss_client_00.key</code> 第 1 套客户端配置 • <code>wss_ca_01.crt</code> • <code>wss_client_01.crt</code> • <code>wss_client_01.key</code>	<i><code>AT+WSCFG</code></i>
HTTP 客户端	第 0 套客户端配置 • <code>https_ca_00.crt</code> • <code>https_client_00.crt</code> • <code>https_client_00.key</code> 第 1 套客户端配置 • <code>https_ca_01.crt</code> • <code>https_client_01.crt</code> • <code>https_client_01.key</code>	<i><code>AT+HTTPCFG</code></i>
Wi-Fi 企业版客户端 (WPA2-Enterprise/WPA3-Enterprise)	• <code>wpa_ent_ca.pem</code> • <code>wpa_ent_client.crt</code> • <code>wpa_ent_client.key</code>	<i><code>AT+CWJEAP</code></i>

请根据自己的需求修改 PKI 配置，然后生成 `mfg_nvs.bin` 文件。

6.8.2 生成 `mfg_nvs.bin` 文件

请参考[生成 `mfg\_nvs.bin`](#)文档生成带有 PKI 配置的 `mfg_nvs.bin`。

6.8.3 下载 mfg_nvs.bin 文件

请参考[下载 mfg_nvs.bin](#) 文档。

6.9 如何自定义低功耗蓝牙服务

本文档介绍了如何利用 ESP-AT 提供的低功耗蓝牙服务源文件在 ESP32-C6 设备上自定义低功耗蓝牙服务。

- [低功耗蓝牙服务源文件介绍](#)
 - [字段规则](#)
- [编译时自定义低功耗蓝牙服务](#)
 - [修改低功耗蓝牙服务源文件](#)
 - [生成 mfg_nvs.bin 文件](#)
 - [下载 mfg_nvs.bin 文件](#)

低功耗蓝牙服务被定义为 GATT 结构的多元数组，该数组至少包含一个属性类型 (attribute type) 为 0x2800 的首要服务 (primary service)。每个服务总是由一个服务定义和几个特征组成。每个特征总是由一个值和可选的描述符组成。更多相关信息请参阅 [《蓝牙核心规范》](#) 中的 Generic Attribute Profile (GATT) 一节。

所有支持低功耗蓝牙的芯片共用同一份 gatts_data.csv 格式。下文说明在 ESP32-C6 上各字段是否生效。

6.9.1 低功耗蓝牙服务源文件介绍

低功耗蓝牙服务源文件是 ESP-AT 工程创建低功耗蓝牙服务所依据的文件，文件位于 [customized_partitions/raw_data/ble_data/gatts_data.csv](#)，内容如下表所示。

index	uuid_len	uuid	perm	val_max_len	val_cur_len	value
0	16	0x2800	0x01	2	2	A002
1	16	0x2803	0x01	1	1	02
2	16	0xC300	0x01	1	1	30
3	16	0x2901	0x11	1	1	30
...	...	...	...	...	...	...

以下内容是对上表的说明。

- perm 字段描述属性权限。当某行的 perm 生效时（见下文规则），它在 ESP-AT 工程中的定义如下所示：

```
/* relate to BTA_GATT_PERM_xxx in bta/bta_gatt_api.h */
/**
 * @brief Attribute permissions
 */
#define ESP_GATT_PERM_READ (1 << 0) /* bit 0 - 0x0001 */ /* relate to BTA_GATT_
→PERM_READ in bta/bta_gatt_api.h */
#define ESP_GATT_PERM_READ_ENCRYPTED (1 << 1) /* bit 1 - 0x0002 */ /* relate_
→to BTA_GATT_PERM_READ_ENCRYPTED in bta/bta_gatt_api.h */
#define ESP_GATT_PERM_READ_ENC_MITM (1 << 2) /* bit 2 - 0x0004 */ /* relate to_
→BTA_GATT_PERM_READ_ENC_MITM in bta/bta_gatt_api.h */
#define ESP_GATT_PERM_WRITE (1 << 4) /* bit 4 - 0x0010 */ /* relate to BTA_
→GATT_PERM_WRITE in bta/bta_gatt_api.h */
#define ESP_GATT_PERM_WRITE_ENCRYPTED (1 << 5) /* bit 5 - 0x0020 */ /* relate_
```

(续下页)

(接上页)

```

→to BTA_GATT_PERM_WRITE_ENCRYPTED in bta/bta_gatt_api.h */
#define ESP_GATT_PERM_WRITE_ENC_MITM (1 << 6) /* bit 6 - 0x0040 */ /* relate_
→to BTA_GATT_PERM_WRITE_ENC_MITM in bta/bta_gatt_api.h */
#define ESP_GATT_PERM_WRITE_SIGNED (1 << 7) /* bit 7 - 0x0080 */ /* relate to_
→BTA_GATT_PERM_WRITE_SIGNED in bta/bta_gatt_api.h */
#define ESP_GATT_PERM_WRITE_SIGNED_MITM (1 << 8) /* bit 8 - 0x0100 */ /* _
→relate to BTA_GATT_PERM_WRITE_SIGNED_MITM in bta/bta_gatt_api.h */
#define ESP_GATT_PERM_READ_AUTHORIZATION (1 << 9) /* bit 9 - 0x0200 */
#define ESP_GATT_PERM_WRITE_AUTHORIZATION (1 << 10) /* bit 10 - 0x0400 */

```

- 上表第一行是服务定义。该行的属性类型为 0x2800（首要服务），其 value A002 表示 16 位服务 UUID 0xA002。
 - 第二行是特征声明。UUID 0x2803 表示该行是特征声明。
 - value: 表示下一行特征值属性的特征属性 (characteristic properties)。该字段为 1 字节 (8 位)，每一位表示是否支持对应属性，1 表示支持，0 表示不支持。
- 在 ESP32-C6 上，ESP-AT 会在内部自动转换该属性值，无需处理其中的差异。
例如，value 为 02 表示下一行特征具备 READ 属性。

位	特征属性
0	BROADCAST
1	READ
2	WRITE WITHOUT RESPONSE
3	WRITE
4	NOTIFY
5	INDICATE
6	AUTHENTICATION SIGNED WRITES
7	EXTENDED PROPERTIES

- 第三行定义该特征的特征值属性。该行的 uuid 是特征 UUID，value 是特征的初始值。
- 第四行定义了特征的描述符（可选）。
- value 字段可以缺省。若缺省（留空），ESP-AT 在初始化时会将该属性自动填充为全 0。例如，默认表中特征 0xC301 一行的 value 为空，则该特征会初始化为全 0。

字段规则

- 对于 0x2803 行，只看 value（下一行特征的属性）。本行的 perm 会被忽略、不生效。
- 对于特征值行，本行的 perm 会被忽略、不生效。若未提供下文所述的第 8 个可选字段，特征能力只来自上一行 0x2803 的 value（特征属性）。
- 若特征属性包含 NOTIFY 或 INDICATE，会自动添加 CCCD (0x2902)。gatts_data.csv 中的 0x2902 行会被跳过、不处理（为兼容 CSV 仍可保留这些行；在 ESP32-C6 上它们只是不生效）。
- 其他描述符（如 0x2901）仍会被添加。若未提供下文所述的第 8 个可选字段，这些描述符行以本行的 perm 为准。

可选第 8 字段

在 ESP32-C6 上，特征值行和描述符行还可以在 value 之后追加一个可选字段。前缀 0x/0X 可写可不写。服务定义行 (0x2800)、特征声明行 (0x2803) 等其他行即使填写该字段，也不会生效。

1. 特征值行：第 8 字段为 32 位十六进制串，表示该特征的完整属性标志。若填写了该字段，则上一行 0x2803 的 value（特征属性）会被忽略、不生效，以本字段为准。
例如，原来的：

```
2,16,0xC300,0x01,1,1,30
```

也可以写成：

```
2,16,0xC300,0x01,1,1,30,0x00020000
```

或：

```
2,16,0xC300,0x01,1,1,30,00020000
```

该字段定义如下（可按位或组合）：

```
#define BLE_GATT_CHR_F_BROADCAST 0x00000001
#define BLE_GATT_CHR_F_READ 0x00000002
#define BLE_GATT_CHR_F_WRITE_NO_RSP 0x00000004
#define BLE_GATT_CHR_F_WRITE 0x00000008
#define BLE_GATT_CHR_F_NOTIFY 0x00000010
#define BLE_GATT_CHR_F_INDICATE 0x00000020
#define BLE_GATT_CHR_F_AUTH_SIGN_WRITE 0x00000040
#define BLE_GATT_CHR_F_RELIABLE_WRITE 0x00000080
#define BLE_GATT_CHR_F_AUX_WRITE 0x00000100
#define BLE_GATT_CHR_F_READ_ENC 0x00000200
#define BLE_GATT_CHR_F_READ_AUTHEN 0x00000400
#define BLE_GATT_CHR_F_READ_AUTHOR 0x00000800
#define BLE_GATT_CHR_F_WRITE_ENC 0x00001000
#define BLE_GATT_CHR_F_WRITE_AUTHEN 0x00002000
#define BLE_GATT_CHR_F_WRITE_AUTHOR 0x00004000
#define BLE_GATT_CHR_F_NOTIFY_INDICATE_ENC 0x00008000
#define BLE_GATT_CHR_F_NOTIFY_INDICATE_AUTHEN 0x00010000
#define BLE_GATT_CHR_F_NOTIFY_INDICATE_AUTHOR 0x00020000
```

2. 描述符行：第 8 字段为 8 位十六进制串，表示该描述符的权限。若填写了该字段，则本行的 perm 会被忽略、不生效，以本字段为准。

例如，原来的：

```
3,16,0x2901,0x11,1,1,30
```

也可以写成：

```
3,16,0x2901,0x11,1,1,30,0x80
```

或：

```
3,16,0x2901,0x11,1,1,30,80
```

该字段定义如下（可按位或组合）：

```
#define BLE_ATT_F_READ 0x01
#define BLE_ATT_F_WRITE 0x02
#define BLE_ATT_F_READ_ENC 0x04
#define BLE_ATT_F_READ_AUTHEN 0x08
#define BLE_ATT_F_READ_AUTHOR 0x10
#define BLE_ATT_F_WRITE_ENC 0x20
#define BLE_ATT_F_WRITE_AUTHEN 0x40
#define BLE_ATT_F_WRITE_AUTHOR 0x80
```

有关 UUID 的更多信息请参考 [蓝牙技术联盟分配符](#)。

如果直接在 ESP32-C6 设备上使用默认源文件，不做任何修改，并建立低功耗蓝牙连接，那么在客户端查询服务器服务后，会得到如下结果。

6.9.2 编译时自定义低功耗蓝牙服务

请根据以下步骤自定义低功耗蓝牙服务。

- [修改低功耗蓝牙服务源文件](#)
- [生成 mfg_nvs.bin 文件](#)
- [下载 mfg_nvs.bin 文件](#)

Unknown Service

UUID: 0000a002-0000-1000-8000-00805f9b34fb

PRIMARY SERVICE

Unknown Characteristic

UUID:

0000c300-0000-1000-8000-00805f9b34fb

Properties: READ

Value: (0x) 30, "0"

Descriptors:

Characteristic User Description

UUID: 0x2901

Value: 0

**修改低功耗蓝牙服务源文件**

可定义多个服务，例如，若要定义三个服务（Server_A、Server_B 和 Server_C），则需要将这三个服务按顺序排列。由于定义每个服务的操作大同小异，这里我们以定义一个服务为例，其他服务你可以按照此例进行定义。

1. 添加服务定义。

本例定义了一个值为 0xFF01 的主要服务。

index	uuid_len	uuid	perm	val_max_len	val_cur_len	value
31	16	0x2800	0x01	2	2	FF01

2. 添加特征说明和特征值。

本例定义了一个 UUID 为 0xC300 的可读可写特征，并将其值设置为 0x30。

声明行 perm 与特征值行 perm 会被忽略、不生效。

index	uuid_len	uuid	perm	val_max_len	val_cur_len	value
32	16	0x2803	0x01	1	1	0A
33	16	0xC300	0x11	1	1	30

3. 添加特征描述符（可选）。

步骤 2 中的特征属性为 0A（表示 READ | WRITE），不需要 CCCD。下方内容是针对支持 NOTIFY 或 INDICATE 的特征的 **独立可选示例**，与上面的 0A 示例无关。若需要 NOTIFY，请相应设置 0x2803 的属性（例如 1A 表示 READ | WRITE | NOTIFY）。

若属性包含 NOTIFY 或 INDICATE，CCCD 会自动添加。无需再添加 0x2902 行；若 CSV 中已有该行，会被跳过、不生效。其他非 0x2902 的描述符仍会添加，且该行的 perm 生效。

完成以上步骤后，自定义的低功耗蓝牙服务可如下定义（0A 的 READ | WRITE 示例；无需 0x2902 行）：

index	uuid_len	uuid	perm	val_max_len	val_cur_len	value
31	16	0x2800	0x01	2	2	FF01
32	16	0x2803	0x01	1	1	0A
33	16	0xC300	0x11	1	1	30

请根据自己的需求修改 GATTS 配置，然后生成 mfg_nvs.bin 文件。

生成 mfg_nvs.bin 文件

请参考[生成 mfg_nvs.bin](#) 文档生成带有低功耗蓝牙的服务配置的 mfg_nvs.bin。

下载 mfg_nvs.bin 文件

请参考[下载 mfg_nvs.bin](#) 文档。

下载完成后，重新建立低功耗蓝牙连接，在客户端查询的服务器服务如下所示。



6.10 如何自定义分区

本文档介绍了如何通过修改 ESP-AT 提供的 at_customize.csv 表来自定义 ESP32-C6 设备中的分区。共有两个分区表：一级分区表和二级分区表。

一级分区表 partitions_at.csv 供系统使用，在此基础上生成 partitions_at.bin 文件。如果一级分区表出错，系统将无法启动。因此，不建议修改 partitions_at.csv。

ESP-AT 提供了二级分区表 at_customize.csv 供你存储自定义数据块。它基于一级分区表。

要修改 ESP32-C6 设备中的分区，请按照前三个步骤操作。第四部分举例说明了前三个步骤。

- [修改 at_customize.csv](#)
- [生成 at_customize.bin](#)
- [烧录 at_customize.bin 至 ESP32-C6 设备](#)
- [示例](#)

6.10.1 修改 at_customize.csv

请参考下表找到模组的 at_customize.csv。

表 1: at_customize.csv 路径

平台	模组	路径
ESP32-C6	ESP32C6-4MB (所有带 4 MB flash 的 ESP32-C6 系列)	module_config/module_esp32c6_default/at_customize.csv

然后，在修改 at_customize.csv 时遵循以下规则。

- 已定义的用户分区的 Name 和 Type 不可更改，但 SubType、Offset 和 Size 可以更改。
- 如果你需要添加一个新的用户分区，请先检查它是否已经在 ESP-IDF (esp_partition.h) 中定义。
 - 如果已定义，请保持 Type 值与 ESP-IDF 的相同。
 - 如果未定义，请将 Type 设置为 0x40。
- 用户分区的 Name 不应超过 16 字节。
- at_customize 分区的默认大小定义在 partitions_at.csv 表中，添加新用户分区时请不要超出范围。

6.10.2 生成 at_customize.bin

修改 at_customize.csv 后，你可以重新编译 ESP-AT 工程或使用 python 脚本 gen_esp32part.py 来生成 at_customize.bin 文件。

如果使用脚本，在 ESP-AT 工程根目录下执行以下命令，并替换 INPUT 和 OUTPUT。

```
python esp-idf/components/partition_table/gen_esp32part.py <INPUT> [OUTPUT]
```

- INPUT 替换为待解析的 at_customize.csv 或二进制文件的路径。
- OUTPUT 替换为生成的二进制或 CSV 文件的路径，如果省略，将使用标准输出。

6.10.3 烧录 at_customize.bin 至 ESP32-C6 设备

将 at_customize.bin 下载到 flash 中。关于如何将二进制文件烧录至 ESP32-C6 设备，请参考[烧录 AT 固件至设备](#)。下表为不同模组 at_customize.bin 文件的下载地址。

表 2: 不同模组 at_customize.bin 的下载地址

平台	模组	地址	大小
ESP32-C6	ESP32C6-4MB (所有带 4 MB flash 的 ESP32-C6 系列)	0x1E000	0x42000

在某些情况下，必须将 at_customize.bin 下载到 flash 后才能使用一些 AT 命令：

- [AT+SYSFLASH](#): 查询或读写 flash 用户分区
- [AT+FS](#): 文件系统操作
- SSL 服务器相关命令
- BLE 服务器相关命令

6.10.4 示例

本节介绍如何将名为 test 的 4 KB 分区添加到 ESP32-C6-MINI-1 模组中。

首先找到 ESP32-C6-MINI-1 的 at_customize.csv 表，设置新分区的 Name、Type、SubType、Offset 和 Size。

```
# Name, Type, SubType, Offset, Size
...
test, 0x40, 15, 0x3E000, 4K
fs_storage, data, 0xff, 0x47000, 100K
```

第二步，重新编译 ESP-AT 工程，或者在 ESP-AT 根目录下执行 python 脚本生成 at_customize.bin。

```
python esp-idf/components/partition_table/gen_esp32part.py -q ./module_config/
↪module_esp32c6_default/at_customize.csv at_customize.bin
```

然后，ESP-AT 根目录中会生成 at_customize.bin。

第三步，下载 at_customize.bin 至 flash。

在 ESP-AT 工程根目录下执行以下命令，并替换 PORT 和 BAUD。

```
python esp-idf/components/esptool_py/esptool/esptool.py -p PORT -b BAUD --before_
↪default_reset --after hard_reset --chip auto write_flash --flash_mode dio --
↪flash_size detect --flash_freq 40m 0x1E000 ./at_customize.bin
```

- PORT 替换为端口名称。
- BAUD 替换为波特率。

6.11 如何增加一个新的模组支持

ESP-AT 工程支持多个模组，并提供了模组的配置文件：[factory_param_data.csv](#) 和 [module_config](#)。如果要在 ESP-AT 工程中添加对某个 ESP32-C6 模组的支持，则需要修改这些配置文件。此处的“ESP32-C6 模组”指的是：

- ESP-AT 工程暂未适配支持的模组，包括 ESP-AT 已适配相应芯片的模组，和未适配相应芯片的模组。但不建议添加后者，因为工作量巨大，此文档也不做阐述。
- ESP-AT 工程已适配支持的模组，但用户需要对其修改默认配置的。

本文档将说明如何在 ESP-AT 工程中为 ESP-AT 已支持的某款 ESP32-C6 芯片添加新的模组支持，下文中以添加对 ESP32-C6-MINI-1 支持为例，该模组使用 SPI 而不是默认的 UART 接口。

- [第一步：配置新增模组的出厂参数](#)
- [第二步：配置新增模组的 OTA](#)
- [第三步：新增模组配置](#)

6.11.1 第一步：配置新增模组的出厂参数

打开本地的 [factory_param_data.csv](#)，在表格最后插入一行，根据实际需要设置相关参数。本例中，我们将 platform 设置为 PLATFORM_ESP32C6、module_name 设置为 ESP32C6-USER-DEFINED，其他参数设置值见下表（参数含义请参考[出厂参数配置介绍](#)）。

- platform: PLATFORM_ESP32C6
- module_name: ESP32C6-USER-DEFINED
- description:
- version: 4
- max_tx_power: 78
- uart_port: 1
- start_channel: 1
- channel_num: 13
- country_code: CN
- uart_baudrate: -1
- uart_tx_pin: -1
- uart_rx_pin: -1
- uart_cts_pin: -1
- uart_rts_pin: -1

6.11.2 第二步：配置新增模组的 OTA

在 `at/src/at_default_config.c` 中的 `s_module_info` 结构体中添加自定义模组的信息。

`s_module_info` 结构体提供 OTA 升级验证 token:

```
typedef struct {
    char* module_name;
    char* ota_token;
    char* ota_ssl_token;
} at_module_info_t;
```

若不想使用 OTA 功能，那么第二个参数 `ota_token` 和第三个参数 `ota_ssl_token` 应该设置为 `NULL`，第一个参数 `module_name` 必须与 `factory_param_data.csv` 文件中的 `module_name` 一致。

下面是修改后的 `s_module_info` 结构体。

```
static const at_module_info_t s_module_info[] = {
#ifdef CONFIG_IDF_TARGET_ESP32
    ...
#endif

#ifdef CONFIG_IDF_TARGET_ESP32C3
    ...
#endif

#ifdef CONFIG_IDF_TARGET_ESP32C2
    ...
#endif

#ifdef CONFIG_IDF_TARGET_ESP32C5
    ...
#endif

#ifdef CONFIG_IDF_TARGET_ESP32C6
    ...
#endif

#ifdef CONFIG_IDF_TARGET_ESP32C6
    {"MY_MODULE", CONFIG_ESP_AT_OTA_TOKEN_MY_MODULE, CONFIG_ESP_AT_OTA_
    ↪SSL_TOKEN_MY_MODULE }, // MY_MODULE
#endif
};
```

宏 `CONFIG_ESP_AT_OTA_TOKEN_MY_MODULE` 和宏 `CONFIG_ESP_AT_OTA_SSL_TOKEN_MY_MODULE` 定义在头文件 `at/private_include/at_ota_token.h` 中。

```
#if defined(CONFIG_IDF_TARGET_ESP32C6)
...
#define CONFIG_ESP_AT_OTA_TOKEN_MY_MODULE CONFIG_ESP_AT_OTA_TOKEN_DEFAULT

...
#define CONFIG_ESP_AT_OTA_SSL_TOKEN_MY_MODULE CONFIG_ESP_AT_OTA_SSL_TOKEN_
    ↪DEFAULT
```

6.11.3 第三步：新增模组配置

下表列出了 ESP-AT 工程支持的平台（即芯片系列）名称、模组配置名称以及各个模组配置对应的配置文件的位置。

平台	模组配置名称	对应的默认配置文件
ESP32	WROOM-32	module_config/module_esp32_default/sdkconfig.defaults module_config/module_esp32_default/sdkconfig_silence.defaults
ESP32	PICO-D4	module_config/module_esp32_default/sdkconfig.defaults module_config/module_esp32_default/sdkconfig_silence.defaults
ESP32	SOLO-1	module_config/module_esp32_default/sdkconfig.defaults module_config/module_esp32_default/sdkconfig_silence.defaults
ESP32	MINI-1	module_config/module_esp32_default/sdkconfig.defaults module_config/module_esp32_default/sdkconfig_silence.defaults
ESP32	WROVER-32	module_config/module_wrover-32/sdkconfig.defaults module_config/module_wrover-32/sdkconfig_silence.defaults
ESP32	ESP32-D2WD	module_config/module_esp32-d2wd/sdkconfig.defaults module_config/module_esp32-d2wd/sdkconfig_silence.defaults
ESP32	ESP32-SDIO	module_config/module_esp32-sdio/sdkconfig.defaults module_config/module_esp32-sdio/sdkconfig_silence.defaults
ESP32-C2	ESP32C2-2MB	module_config/module_esp32c2-2mb/sdkconfig.defaults module_config/module_esp32c2-2mb/sdkconfig_silence.defaults
ESP32-C2	ESP32C2-2MB-BLE	module_config/module_esp32c2-2mb-ble/sdkconfig.defaults module_config/module_esp32c2-2mb-ble/sdkconfig_silence.defaults
ESP32-C2	ESP32C2-4MB	module_config/module_esp32c2_default/sdkconfig.defaults module_config/module_esp32c2_default/sdkconfig_silence.defaults
ESP32-C3	MINI-1	module_config/module_esp32c3_default/sdkconfig.defaults module_config/module_esp32c3_default/sdkconfig_silence.defaults
ESP32-C3	ESP32C3-SPI	module_config/module_esp32c3-spi/sdkconfig.defaults module_config/module_esp32c3-spi/sdkconfig_silence.defaults
ESP32-C3	ESP32C3_RAINM	module_config/module_esp32c3_rainmaker/sdkconfig.defaults module_config/module_esp32c3_rainmaker/sdkconfig_silence.defaults
ESP32-C5	ESP32C5-4MB	module_config/module_esp32c5_default/sdkconfig.defaults module_config/module_esp32c5_default/sdkconfig_silence.defaults
ESP32-C6	ESP32C6-4MB	module_config/module_esp32c6_default/sdkconfig.defaults module_config/module_esp32c6_default/sdkconfig_silence.defaults
ESP32-C61	ESP32C61-4MB	module_config/module_esp32c61_default/sdkconfig.defaults module_config/module_esp32c61_default/sdkconfig_silence.defaults

注意:

- 当 `python build.py install` 中的 `silence mode` 为 0 时, 模组配置对应的配置文件为 `sdkconfig.defaults`。
- 当 `python build.py install` 中的 `silence mode` 为 1 时, 模组配置对应的配置文件为 `sdkconfig_silence.defaults`。

首先, 进入 `module_config` 文件夹, 创建一个子文件夹来存放模组配置的配置文件 (文件夹名称为小写), 然后在其中加入配置文件 `IDF_VERSION`、`patch`、`at_customize.csv`、`partitions_at.csv`、`sdkconfig.defaults` 以及 `sdkconfig_silence.defaults`。

本例中, 我们复制粘贴 `module_esp32c6_default` 文件夹及其中的配置文件, 并重命名为 `module_esp32c6-user-defined`。在本例中, 配置文件 `IDF_VERSION`、`patch`、`at_customize.csv` 和 `partitions_at.csv` 无需修改, 我们只需修改 `sdkconfig.defaults` 和 `sdkconfig_silence.defaults`:

- 使用 `module_esp32c6-user-defined` 文件夹下的分区表, 需要修改如下配置

```
CONFIG_PARTITION_TABLE_CUSTOM_FILENAME="module_config/module_esp32c6-user-
→defined/partitions_at.csv"
CONFIG_PARTITION_TABLE_FILENAME="module_config/module_esp32c6-user-defined/
→partitions_at.csv"
CONFIG_AT_CUSTOMIZED_PARTITION_TABLE_FILE="module_config/module_esp32c6-user-
→defined/at_customize.csv"
```

- 使用 SPI 配置, 移除 UART 配置
 - 移除 UART 配置

```
CONFIG_AT_BASE_ON_UART=n
```

- 新增 SPI 配置

```
CONFIG_AT_BASE_ON_SPI=y
CONFIG_SPI_STANDARD_MODE=y
CONFIG_SPI_SCLK_PIN=6
CONFIG_SPI_MOSI_PIN=7
CONFIG_SPI_MISO_PIN=2
CONFIG_SPI_CS_PIN=10
CONFIG_SPI_HANDSHAKE_PIN=3
CONFIG_SPI_NUM=1
CONFIG_SPI_MODE=0
CONFIG_TX_STREAM_BUFFER_SIZE=4096
CONFIG_RX_STREAM_BUFFER_SIZE=4096
```

完成上述步骤后, 可重新编译 ESP-AT 工程生成模组固件。本例中, 我们在本地编译 AT 固件 [第三步安装环境](#) 时, 就可以选择 `PLATFORM_ESP32C6` 和 `ESP32C6-USER-DEFINED` 来生成 AT 固件。

6.12 ESP32-C6 SDIO AT 指南

6.12.1 简介

ESP32-C6 SDIO AT 使用 SDIO 协议进行通讯, 其中 ESP32-C6 作为 SDIO slave 与 MCU 进行通信。

SDIO 可使用一线或四线模式。至少需要 4 根线: `CMD`、`CLK`、`DAT0` 和 `DAT1`。

- 对于一线模式, 至少需要 4 根线: `CMD`、`CLK`、`DAT0` 和 `DAT1`, 其中 `DAT1` 作为中断线;
- 对于四线模式, 需要增加 `DAT2` 和 `DAT3`。

SDIO slave 管脚如下所示:

- CLK is GPIO19
- CMD is GPIO18
- DAT0 is GPIO20
- DAT1 is GPIO21
- DAT2 is GPIO22 (四线)
- DAT3 is GPIO23 (四线)

6.12.2 如何使用 SDIO AT

在测试 SDIO AT 通信之前, 首先请参照 [SD Pull-up Requirements](#) 的介绍对 ESP32-C6 硬件进行处理, 否则 SDIO 通信将会异常。

Slave 侧

AT 项目默认使用 UART 作为传输介质, 你可以通过 `./build.py menuconfig>Component config>AT>communicate method for AT command>AT through SDIO` 切换为用 SDIO 作为传输介质。之后重新编译 esp-at 工程, 烧录新的固件并运行。

Host 侧

ESP-AT 提供了 ESP32-C6 和 STM32 作为 SDIO host 的 [at_sdio_host](#) 参考示例。对于 ESP32-C6 的参考示例, 可以使用 AT 工程的 esp-idf 版本进行编译烧录; 对于 STM32, 我们提供的方案基于 STM32F103ZET, 请使用 Keil5 进行编译烧录。

对于其他平台, 可以参照这两个示例进行适配。

6.12.3 SDIO 交互流程

Host 侧

1. 初始化 SDIO host
 - SDIO host 初始化主要是 SDIO 协议的初始化, 包括设置 1 线或者 4 线、SDIO 频率、初始化 SD mode。
2. 协商 SDIO 通讯
 - 这部分主要按照 SDIO spec 协议标准的要求, 跟 SDIO slave 协商参数。
 - 特别需要注意的是, 如果 SDIO host 和 slave 同时重启, 那么, 协商需要等待 slave 初始化完成后开始。一般 host 会在启动时添加延时, 等待 slave 启动完成, 再开始协商 SDIO 通信。
3. 接收数据
 - 接收数据主要依靠监测 DAT1 的中断信号。当接收到中断信号后, host 读取中断源并判断中断信号, 如果中断是 slave 有数据要发送, host 会调用 CMD53 读取 slave 的数据。
4. 发送数据
 - SDIO AT DEMO 中, 发送数据通过串口输入, 然后 host 调用 CMD53 将数据发送过去。
 - 需要注意的是, 如果发送超时, 那么有可能 slave 侧出现异常, 此时 host 和 slave 需要重新初始化 SDIO。
 - 在调用 `AT+RST` 或者 `AT+RESTORE` 命令后, host 和 slave 也同样需要重新初始化 SDIO。

Slave 侧

SDIO slave 的处理与 SDIO host 类似, slave 在接收到 SDIO host 发送的数据后, 通知 AT core 并将数据发送给 AT core 进行处理, 在 AT core 处理完成后, 再发送数据给 SDIO host。

1. 初始化 SDIO slave
 - 调用 `sdio_slave_initialize` 初始化 SDIO slave driver。
 - 调用 `sdio_slave_recv_register_buf` 注册接收用的 buffer, 为了加快接收速度, 此处注册了多个接收 buffer。

- 调用 `sdio_slave_recv_load_buf` 加载刚刚注册的 `buffer`，准备接收数据。
 - `sdio_slave_set_host_intena` 用于设置 `host` 可用中断，主要用到的是新数据包发送中断 `SDIO_SLAVE_HOSTINT_SEND_NEW_PACKET`。
 - 调用 `sdio_slave_start` 在硬件上开始接收和发送。
2. 发送数据
 - 因为 `SDIO slave` 发送的数据需要保证能被 `DMA` 访问，所以需要先把 `AT` 中的数据拷贝到可被 `DMA` 访问的内存中，然后调用 `sdio_slave_transmit` 进行发送。
 3. 接收数据
 - 为了优化接收 `SDIO` 数据传输给 `AT core` 的速率，在调用 `sdio_slave_recv` 接收 `SDIO` 数据后，使用了循环链表将接收到的数据传输到 `AT core`。

6.13 SPI AT 指南

本文档主要介绍 `SPI AT` 的实现与使用，主要涉及以下几个方面：

- [简介](#)
- [使用 `SPI AT`](#)
- [SPI AT 速率](#)

6.13.1 简介

`SPI AT` 基于 `AT` 工程，使用 `SPI` 协议进行数据通信。在使用 `SPI AT` 时，`MCU` 设备作为 `SPI master`，`ESP32-C6` 设备作为 `SPI slave`，通信双方通过 `SPI` 协议实现基于 `AT` 命令的数据交互。

使用 `SPI AT` 的优势

`AT` 工程默认使用 `UART` 协议进行数据通信，但是 `UART` 协议在一些需要高速传输数据的应用场景并不适用，因此，使用支持更高传输速率的 `SPI` 协议传输数据成为一种较好的选择。

如何启用 `SPI AT`？

你可以通过下述步骤配置并启用 `SPI AT`：

1. 通过 `./build.py menuconfig > Component config > AT > communicate method for AT command > AT through SPI` 使能 `SPI AT`。
2. 通过 `./build.py menuconfig > Component config > AT > communicate method for AT command > AT SPI Data Transmission Mode` 选择 `SPI` 数据传输模式。
3. 通过 `./build.py menuconfig > Component config > AT > communicate method for AT command > AT SPI GPIO settings` 配置 `SPI` 使用的 `GPIO` 管脚。
4. 通过 `./build.py menuconfig > Component config > AT > communicate method for AT command > AT SPI driver settings` 选择 `SPI` 从机的工作模式，并配置相关缓存区的大小。
5. 重新编译 `esp-at` 工程（参考[本地编译 `ESP-AT` 工程](#)），烧录新的固件并运行。

`SPI AT` 默认管脚

下表给出了不同系列的 `ESP32-C6` 设备使用 `SPI AT` 时默认的硬件管脚：

表 3: SPI AT 默认管脚

信号	GPIO 编号
SCLK	19
MISO	20
MOSI	18
CS	23
HANDSHAKE	21
GND	GND
QUADWP (qio/qout) ¹	22
QUADHD (qio/qout) ¹	2

说明 1: QUADWP 引脚和 QUADHD 引脚仅在使用 4 线 SPI 工作时使用。

你可以通过 `./build.py menuconfig>Component config>AT>communicate method for AT command>AT through SPI>AT SPI GPIO settings`, 然后编译工程来配置 SPI AT 对应的管脚 (参考本地编译 [ESP-AT 工程](#))。

6.13.2 使用 SPI AT

在使用 SPI AT 时, ESP32-C6 设备上运行的 SPI slave 工作在半双工通信模式下。

握手线 (handshake line)

SPI 是一种 master-slave 结构的外设, 所有的传输均由 master 发起, slave 无法主动传输数据。但是, 使用 AT 命令进行数据交互时, 需要 ESP32-C6 设备主动能够主动上报一些信息。因此, 我们在 SPI master 和 slave 之间添加了一个握手线, 来实现 slave 主动向 master 上报信息的功能。

当 slave 需要传输数据时, 将会把握手管脚主动拉高, 这会在 master 侧产生一个上升沿的 GPIO 中断, master 发起与 slave 的通信, 传输完成后, slave 将握手管脚拉低, 结束此次通信。

使用握手线的具体方法为:

- Master 向 slave 发送 AT 数据时, 使用握手线的方法为:
 - master 向 slave 发送请求传输数据的请求, 然后等待 slave 向握手线发出的允许发送数据的信号。
 - master 检测到握手线上的 slave 发出的允许发送的信号后, 开始发送数据。
 - master 发送数据后, 通知 slave 数据发送结束。
 如果在 slave 接收数据期间, master 提交发送请求, 请求会被缓冲并在接收完成后处理, 避免产生两次上升沿中断。这确保了数据处理的稳定性。
- Master 接收 slave 发送的 AT 数据时, 使用握手线的方法为:
 - slave 通过握手线通知 master 开始接收数据。
 - master 接收数据, 并在接收所有数据后, 通知 slave 数据已经全部接收。

通信格式

SPI AT 的通信格式为 CMD+ADDR+DUMMY+DATA (读/写)。在使用 SPI AT 时, SPI master 使用到的一些通信报文介绍如下:

- Master 向 slave 发送数据时的通信报文:

表 4: 主机向从机发送数据

CMD (1 字节)	ADDR (1 字节)	DUMMY (1 字节)	DATA (高达 4092 字节)
0x3	0x0	0x0	data_buffer

- Master 向 slave 发送数据结束后，需要发送一条通知消息来结束本次传输，具体的通信报文为：

表 5: 主机通知从机，主机发送数据已经结束

CMD (1 字节)	ADDR (1 字节)	DUMMY (1 字节)	DATA
0x7	0x0	0x0	null

- Master 接收 slave 发送的数据时的通信报文：

表 6: 主机读取从机发送的数据

CMD (1 字节)	ADDR (1 字节)	DUMMY (1 字节)	DATA (高达 4092 字节)
0x4	0x0	0x0	data_buffer

- Master 接收 slave 发送的数据后，需要发送一条通知消息来结束本次传输，具体的通信报文为：

表 7: 主机通知从机，主机读取数据已经结束

CMD (1 字节)	ADDR (1 字节)	DUMMY (1 字节)	DATA
0x8	0x0	0x0	null

- Master 向 slave 发送请求传输指定大小数据的通信报文：

表 8: 主机向从机发送写数据请求

CMD (1 字节)	ADDR (1 字节)	DUMMY (1 字节)	DATA (4 字节)
0x1	0x0	0x0	data_info

其中 4 字节的 data_info 中包含了本次请求传输数据的数据包信息，具体格式如下：

- Master 向 slave 发送的数据的字节数，长度 0 ~ 15 bit。
- Master 向 slave 发送的数据包的序列号，该序列号在 master 每次发送时递增，长度 16 ~ 23 bit。
- Magic 值，长度 24 ~ 31 bit，固定为 0xFE。

字节序：上述 4 字节 DATA 在总线上按 **小端 (little-endian)** 传输（字节 0 = bit 0 ~ 7，字节 1 = bit 8 ~ 15，字节 2 = bit 16 ~ 23，字节 3 = bit 24 ~ 31）。在大端主机上实现时需按此约定打包/解包。

- Master 检测到握手线上有 slave 发出的信号后，需要发送一条消息查询 slave 进入接收数据的工作模式，还是进入到发送数据的工作模式，具体的通信报文为：

表 9: 主机发送请求，查询从机的可读/可写状态

CMD (1 字节)	ADDR (1 字节)	DUMMY (1 字节)	DATA (4 字节)
0x2	0x4	0x0	slave_status

发送查询请求后，slave 返回的状态信息将存储在 4 字节的 slave_status 中，其具体的格式如下：

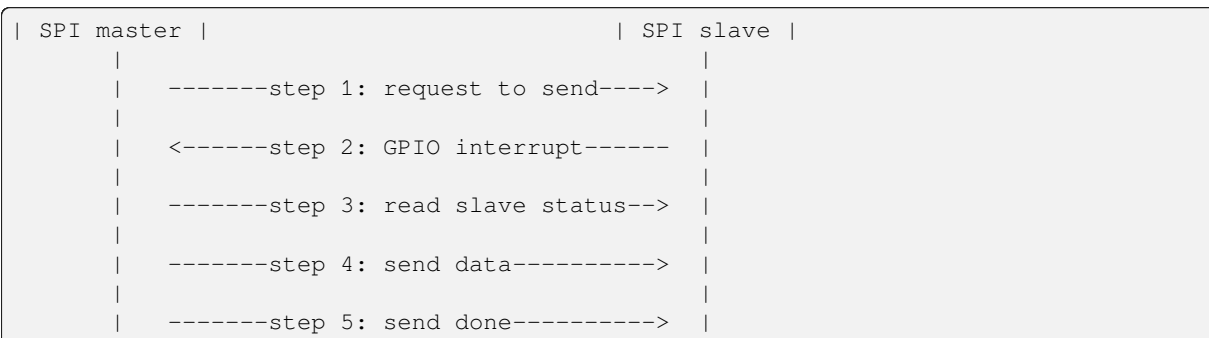
- slave 需要向 master 发送的数据的字节数，长度 0 ~ 15 bit；仅当 slave 处于可读状态时，该字段数字有效。
- 数据包序列号，长度 16 ~ 23 bit，当序列号达到最大值 0xFF 时，下一个数据包的序列号重新设置为 0x0。当 slave 处于可写状态时，该字段为 master 需向 slave 发送的下一个数据包的序列号；当 slave 处于可读状态时，该字段为 slave 向 master 发送的下一个数据包的序列号。
- slave 的可读/可写状态，长度 24 ~ 31 bit，其中，0x1 代表可读，0x2 代表可写。

字节序：同上，为小端（字节 0 = bit 0 ~ 7，字节 1 = bit 8 ~ 15，字节 2 = bit 16 ~ 23，字节 3 = bit 24 ~ 31）。

SPI AT 数据交互流程

SPI AT 数据交互流程主要分为两个方面：

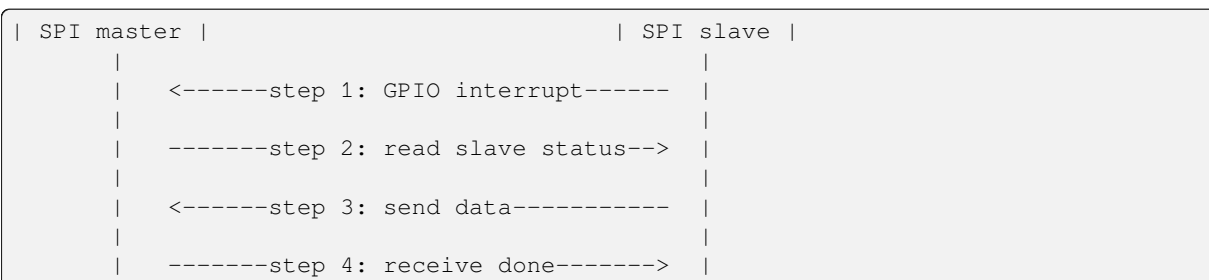
- SPI master 向 slave 发送 AT 命令：



每个步骤具体的说明如下：

1. master 向 slave 发送请求向 slave 写数据的请求。
2. slave 接收 master 的发送请求，若此时 slave 允许接收数据，则向 slave_status 寄存器写入允许 master 写入的标志位，然后通过握手线触发 master 上的 GPIO 中断。
3. master 接收到中断后，读取 slave 的 slave_status 寄存器，检测到 slave 进入接收数据的状态。
4. master 开始向 slave 发送数据。
5. 发送数据结束后，master 向 slave 发送一条代表发送结束的消息。

- SPI slave 向 master 发送 AT 响应：



每个步骤具体的说明如下：

1. slave 向 slave_status 寄存器写入允许 master 读取来自 slave 的数据的标志位，然后通过握手线触发 master 上的 GPIO 中断。
2. master 接收到中断后，读取 slave 的 slave_status 寄存器，检测到 slave 进入发送数据的状态。
3. master 开始接收来自 slave 的数据。
4. 数据接收完毕后，master 向 slave 发送一条代表接收数据结束的消息。

重要： 我们以发送 AT 命令为例，提供了完整的交互流程和逻辑分析仪数据，详见 [at_spi_master/spi/esp32_c_series/README.md](https://github.com/espressif/at_spi_master/tree/master/spi/esp32_c_series/README.md)。强烈建议你阅读并理解相关内容。

SPI AT 对应的 SPI master 侧示例代码

SPI AT 本身是作为 SPI slave 使用的，使用 SPI master 与 SPI slave 进行通信的示例代码请参考 [at_spi_master/spi/esp32_c_series](https://github.com/espressif/at_spi_master/tree/master/spi/esp32_c_series)。

说明 1. 在使用 MCU 开发之前，强烈建议使用 ESP32-C3 或者 ESP32 模拟 MCU 作为 SPI master 来运行此示例，以方便在出现问题时更容易调试问题。

6.13.3 SPI AT 速率

测试说明

- 使用 ESP32 或者 ES32-C3 开发板作为 SPI master, 运行 ESP-AT 中的 `at_spi_master/spi/esp32_c_series` 目录的代码。其软硬件配置如下：
 1. 硬件：CPU 工作频率设置为 240 MHz, flash SPI mode 配置为 QIO 模式, flash 频率设置为 40 MHz。
 2. 软件：基于 ESP-IDF v4.3 的编译环境, 并将示例代码中的 `streambuffer` 的大小调整为 8192 字节。
- 使用 ESP32-C6 作为 SPI slave, 编译并烧录 SPI AT 固件 (参考本地编译 ESP-AT 工程), 并将 ESP32-C6 配置工作在 TCP 透传模式。其软硬件配置如下：
 1. 硬件：CPU 工作频率设置为 160 MHz。
 2. 软件：SPI-AT 的实现代码中, 将 `streambuffer` 的大小设置为 8192 字节, 并使用 ESP-IDF 下的 `example/wifi/ipperf` 中的 `sdkconfig.defaults.esp32c3` 中的相关配置参数。

测试结果

下表显示了我们在屏蔽箱中得到的通信速率结果：

表 10: SPI AT Wi-Fi TCP 通信速率

Clock	SPI mode	master->slave	slave->master
10 M	Standard	0.95 MByte/s	1.00 MByte/s
10 M	Dual	1.37 MByte/s	1.29 MByte/s
10 M	Quad	1.43 MByte/s	1.31 MByte/s
20 M	Standard	1.41 MByte/s	1.30 MByte/s
20 M	Dual	1.39 MByte/s	1.30 MByte/s
20 M	Quad	1.39 MByte/s	1.30 MByte/s
40 M	Standard	1.37 MByte/s	1.30 MByte/s
40 M	Dual	1.40 MByte/s	1.31 MByte/s
40 M	Quad	1.48 MByte/s	1.31 MByte/s

说明：当 SPI 的时钟频率较高时, 受限于上层网络组件的限制, 使用 Dual 或者 Quad 工作模式的通信速率想比较于 Standard 模式并未显著改善。

说明：更多关于 SPI 通信的介绍请参考对应模组的 [技术参考手册](#)。

6.14 如何实现 OTA 升级

本文档指导如何为 ESP32-C6 系列模块实现 OTA 升级。目前, ESP-AT 针对不同场景提供了以下三种 OTA 命令。你可以根据自己的需求选择适合的 OTA 命令。

1. [AT+USEROTA](#)
2. [AT+CIUPDATE](#)
3. [AT+WEBSERVER](#)

文档结构如下所示：

- [OTA 命令对比及应用场景](#)
- [使用 ESP-AT OTA 命令执行 OTA 升级](#)

6.14.1 OTA 命令对比及应用场景

AT+USEROTA

此命令通过 URL 实现 OTA 升级。你可以升级到放置在 HTTP 服务器上的固件。目前该命令仅支持应用分区升级。请参考[AT+USEROTA](#) 获取更多信息。

由于此命令属于用户自定义命令，你可以通过修改 `at/src/at_user_cmd.c` 源代码来实现该命令。

此命令的应用场景如下：

1. 你有属于自己的 HTTP 服务器。
2. 你必须指定 URL。

重要：

- 如果你升级的固件为非乐鑫正式发布的固件，在升级完成后，你可能无法使用 AT+CIUPDATE 命令升级，除非你按照[使用 AT+CIUPDATE 进行 OTA 升级](#)创建了自己的设备。
-

AT+CIUPDATE

此命令使用 `iot.espressif.cn` 作为默认 HTTP 服务器。该命令不仅可以升级应用程序分区，还可以升级 `at_customize.csv` 中定义的用户自定义分区。如果你使用的是乐鑫发布的版本，该命令将只会升级到乐鑫发布的版本。请参考[AT+CIUPDATE](#) 获取更多信息。

使用该命令升级自定义 bin 文件，请选择以下方式之一。

1. 将 `iot.espressif.cn` 替换为你自己的 HTTP 服务器，并实现交互流程。如何实现自己的 AT+CIUPDATE 命令，请参考 `at/src/at_ota_cmd.c`。
2. 在 `iot.espressif.cn` 上创建一个设备，并在其上上传 bin 文件。（前提是模组中运行的固件已经对应你在乐鑫服务器上创建的设备。）请参考[使用 AT+CIUPDATE 进行 OTA 升级](#) 获取更多信息。

此命令的应用场景如下：

1. 你只使用乐鑫发布的固件，只想升级到乐鑫发布的固件。
2. 你希望升级自定义的 bin 文件，但没有自己的 HTTP 服务器。
3. 你有自己的 HTTP 服务器。除了升级应用程序分区外，还希望升级在 `at_customize.csv` 文件中定义的用户自定义分区。

AT+WEBSERVER

此命令通过浏览器或微信小程序升级 AT 固件。目前，该命令仅提供升级应用程序分区的功能。在开始升级之前，请启用 web 服务器命令并提前将 AT 固件复制到电脑或者手机上。你可以参考[AT+WEBSERVER](#) 或者[Web Server AT 示例](#) 获取更多信息。

为了实现你自己的 HTML 页面，请参考示例 `fs_image/index.html`。为了实现你自己的 AT+WEBSERVER 命令，请参考示例 `at/src/at_web_server_cmd.c`。

此命令的应用场景如下：

1. 你需要更方便快捷的 OTA 升级，不依赖于网络状态。

重要：

- 如果你升级的固件为非乐鑫正式发布的固件，在升级完成后，你可能无法使用 AT+CIUPDATE 命令升级，除非你按照[使用 AT+CIUPDATE 进行 OTA 升级](#)创建了自己的设备。
-

6.14.2 使用 ESP-AT OTA 命令执行 OTA 升级

使用 AT+USEROTA 进行 OTA 升级

请参考 [AT+USEROTA](#) 获取更多信息。

使用 AT+CIUPDATE 进行 OTA 升级

通过 [AT+CIUPDATE](#) 命令升级自定义的 bin 文件，首先要做的就是将 bin 文件上传到 [iot.espressif.cn](#) 并且获取到 **token** 值。以下步骤描述了如何在 [iot.espressif.cn](#) 上创建设备并上传 bin 文件。

1. 打开网站 <http://iot.espressif.cn> 或者 <https://iot.espressif.cn>。



图 21: 打开 iot.espressif.cn 网站

2. 点击网页右上角的“Join”，输入你的名字，邮箱地址和密码。

 A screenshot of the 'Join' registration page on the iot.espressif.cn website. The page has a header with the 'IoT-Espressif' logo and navigation links for 'Start', 'Join', and 'Login'. The main heading is 'Join'. Below it, there are three input fields: 'Name' with a placeholder 'Username [a-zA-Z0-9_]+', 'Email' with a placeholder 'Email', and 'Password' with a placeholder 'Password'. At the bottom of the form is a 'Join' button.

图 22: 加入 iot.espressif.cn

备注:

- 当前 Join 功能暂不对新用户开放。如果你想使用该功能，请联系 [乐鑫](#)。

3. 点击网页左上角的“Device”，然后点击“Create”来创建一个设备。

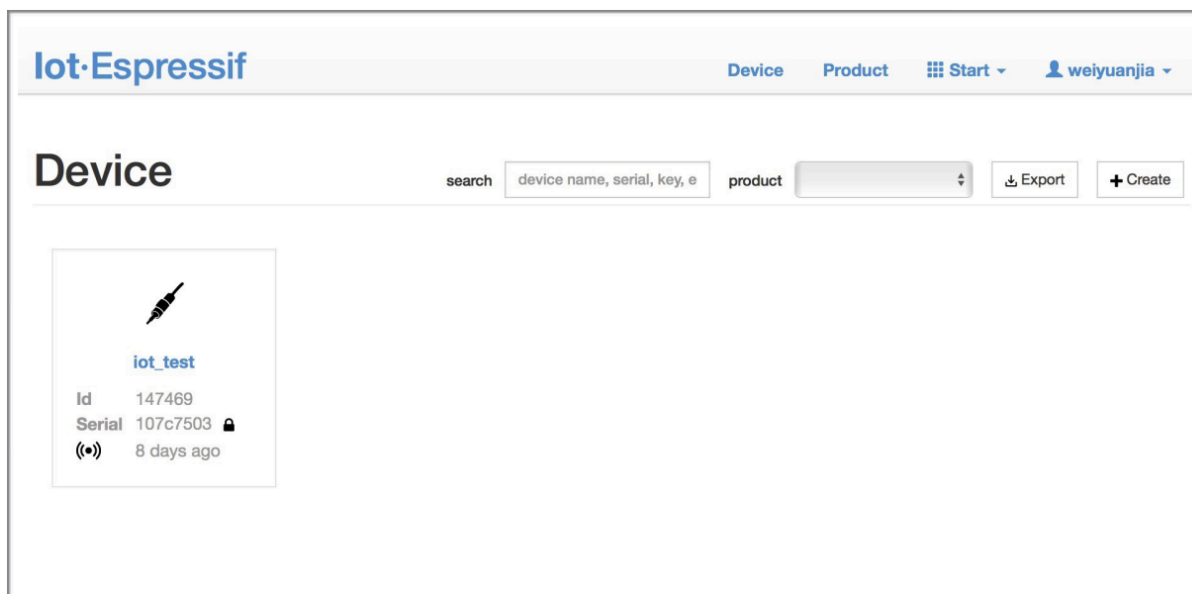


图 23: 点击“Device”

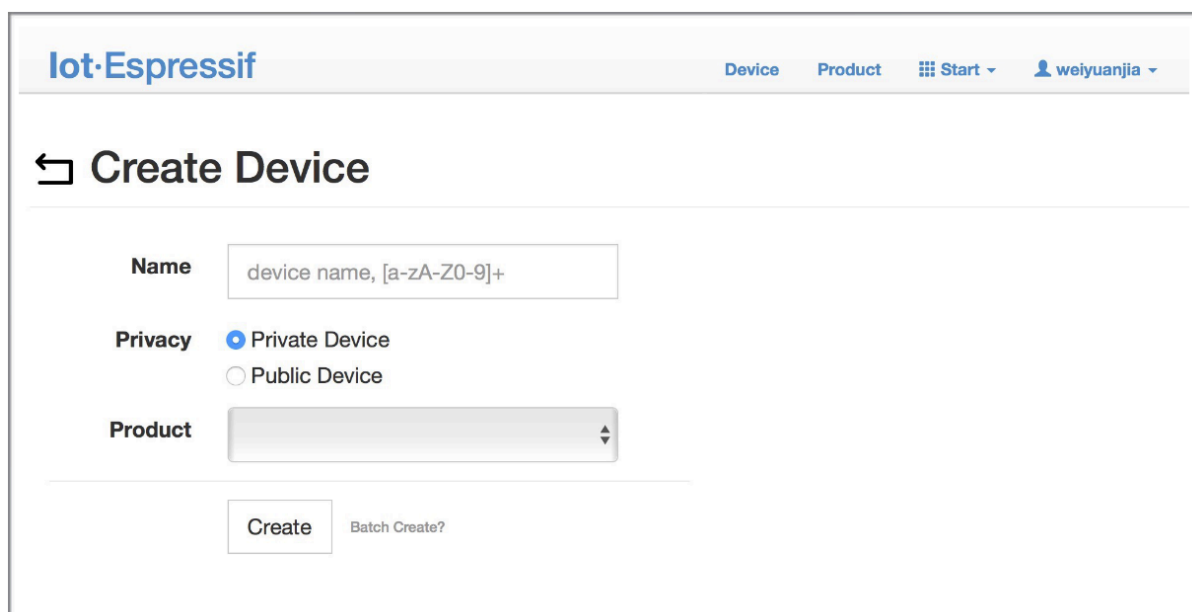


图 24: 点击“Create”

4. 当设备创建成功后会生成一个密钥，如下图所示：
5. 使用该密钥来编译你的 OTA bin 文件。配置 AT OTA token 密钥的过程如下如所示：

备注： 如果使用 SSL OTA，选项“The SSL token for AT OTA”也需要配置。

6. 点击“Product”进入网页，如下如所示。单击创建的设备，在“ROM Deploy”下输入版本和 corename。将步骤 5 中的 bin 文件重命令为“ota.bin”并保存。

备注：

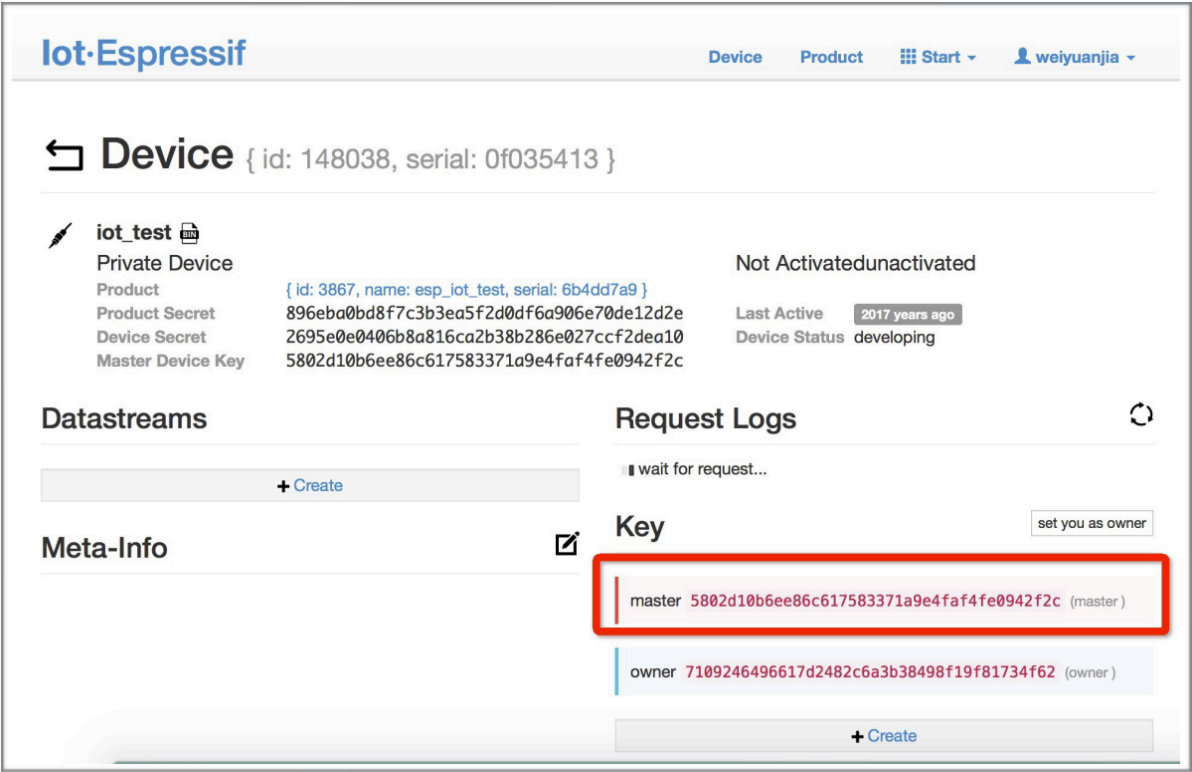


图 25: 生成一个密钥

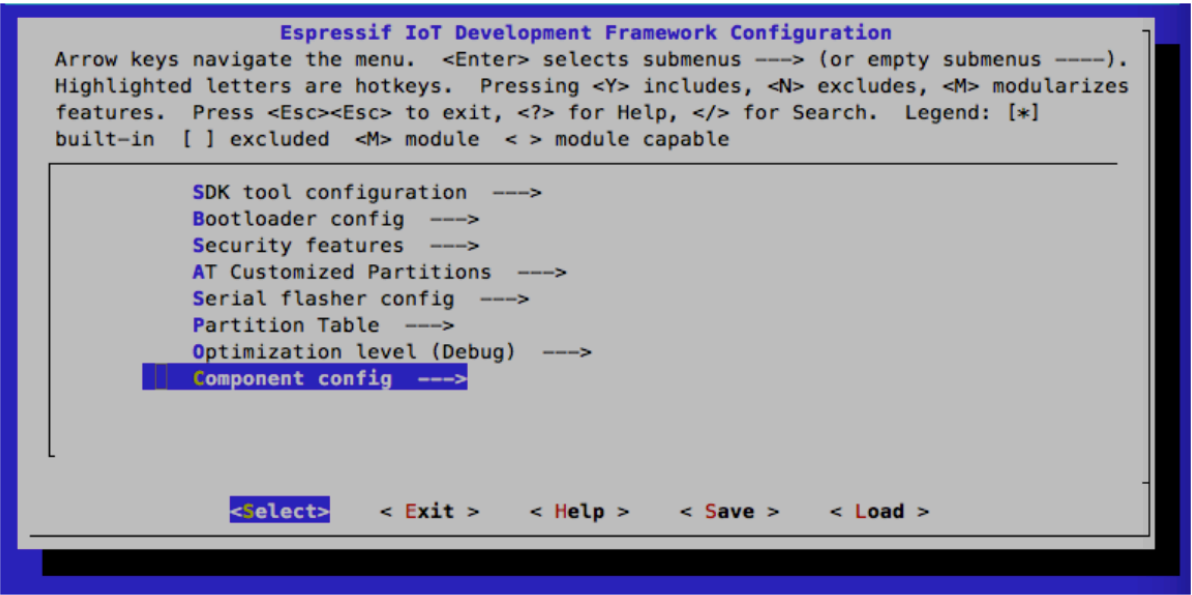


图 26: 配置 AT OTA token 密钥 - 步骤 1

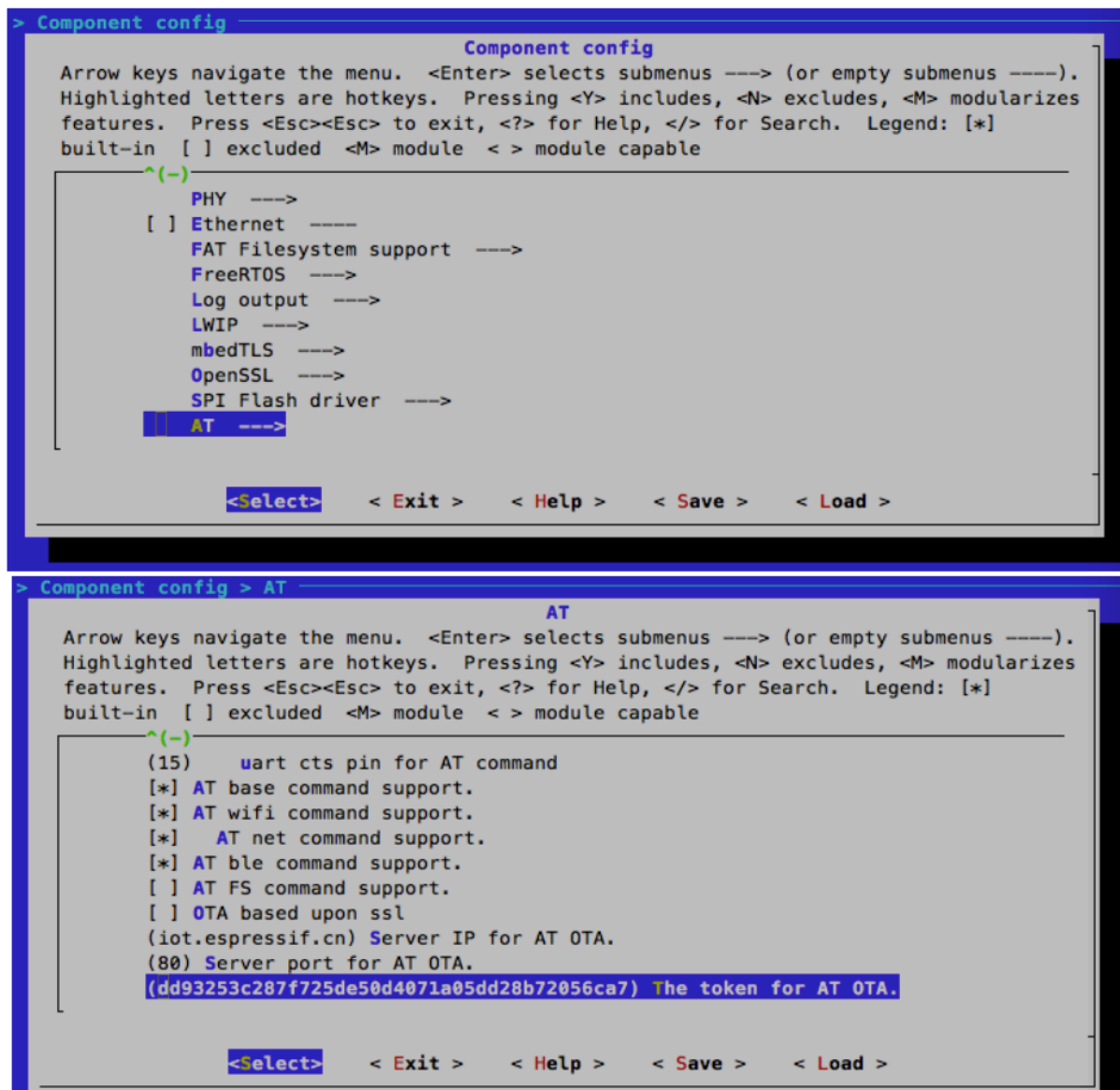


图 27: 配置 AT OTA token 密钥 - 步骤 2 和 3

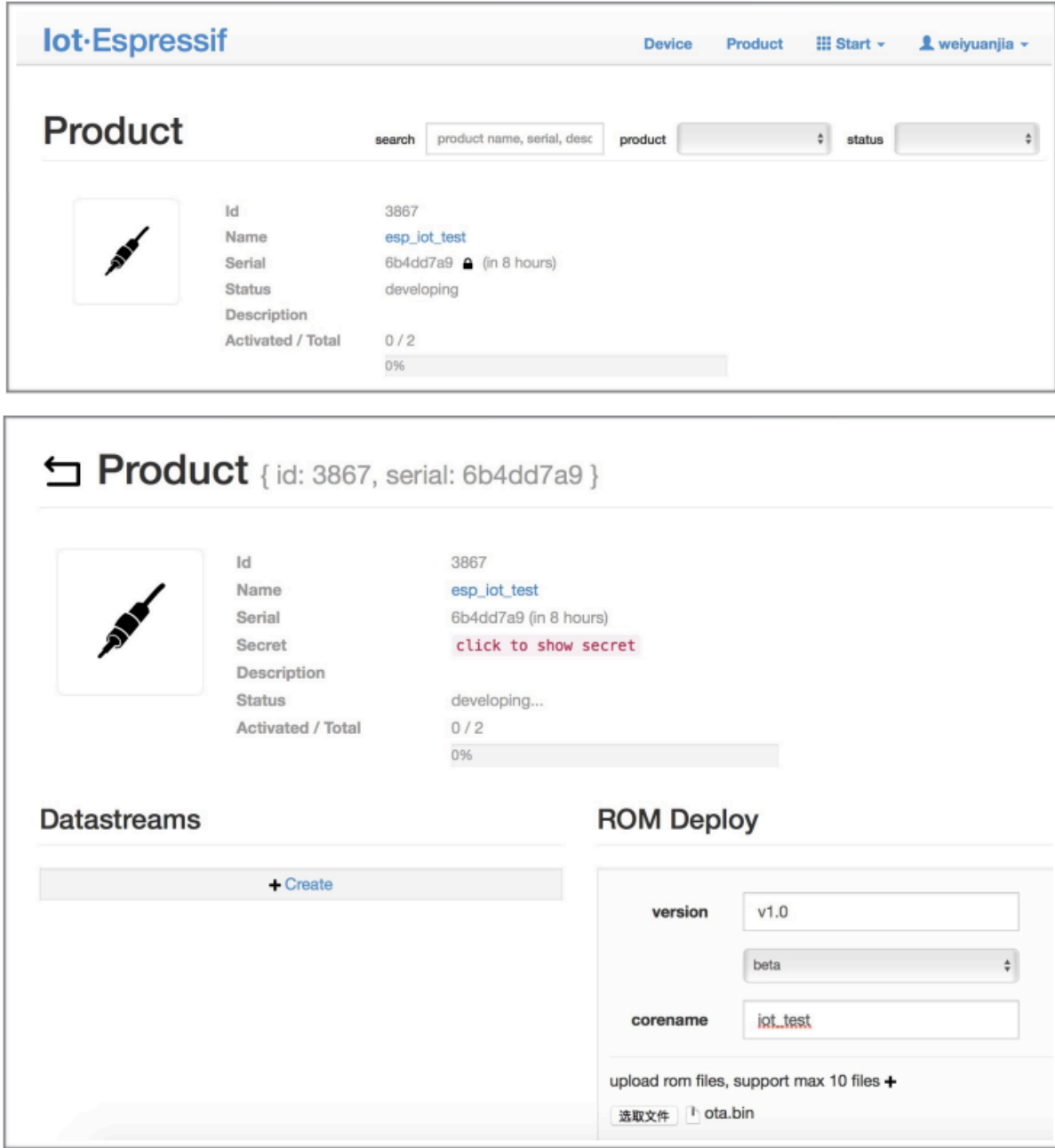


图 28: 输入版本和 corename

- 如果你想要升级 `at_customize.csv` 中定义的用户自定义分区，只需将 `ota.bin` 替换为用户自定义分区的 `bin` 文件即可。
- 对于 `corename` 字段，此字段仅仅用于帮助你区分 `bin` 文件。

7. 单击 `ota.bin` 将其保存为当前版本。

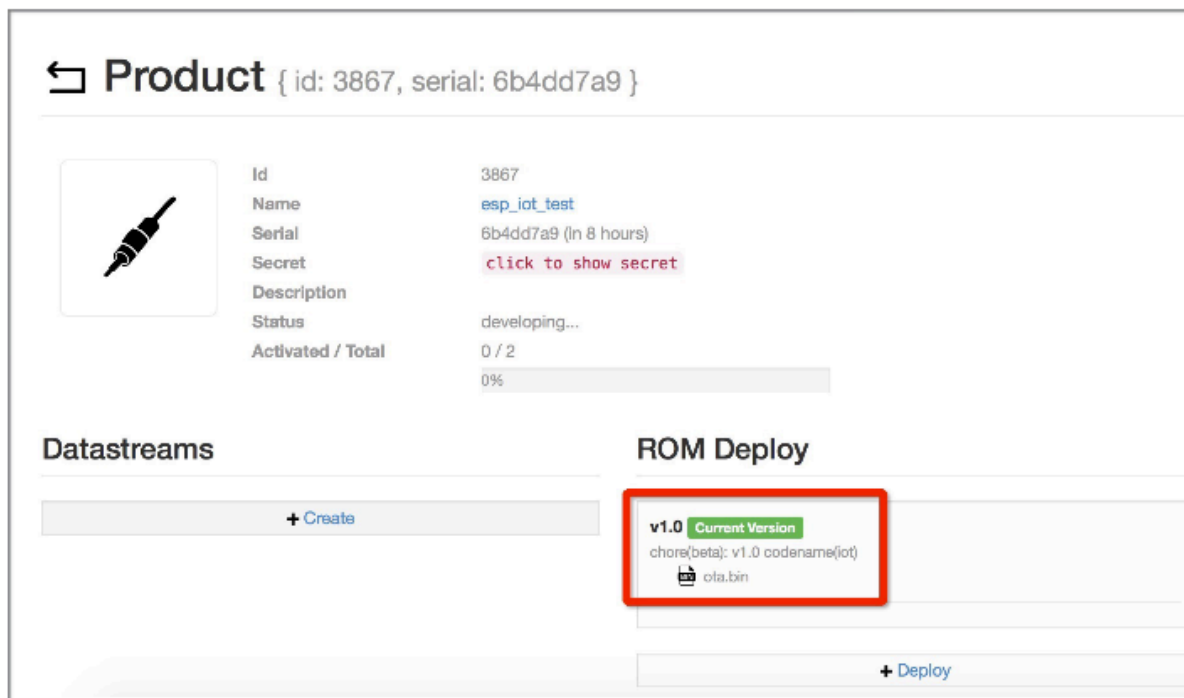


图 29: 保存当前版本的 `ota.bin`

8. 在设备上运行 `AT+CIUPDATE` 命令。如果网络已连接，将开始 OTA 升级。

重要:

- 设置上传到 `iot.espressif.cn` 的 `bin` 文件名称时，请遵循以下规则：
 - 如果升级 `app` 分区，请将 `bin` 文件名设置为 `ota.bin`。
 - 如果升级用户自定义的分区，请将 `bin` 文件名设置为 `at_customize.csv` 中的 `Name` 字段。例如，如果升级 `factory_Param` 分区，请将其设置为 `factory_param.bin`。
- ESP-AT 将新固件存储在备用 OTA 分区中。这样，即使 OTA 由于意外原因失败，原始 ESP-AT 固件也能正常运行。但对于自定义分区，由于 ESP-AT 没有备份措施，请小心升级。
- 如果你打算从一开始只升级自定义的 `bin` 文件，那么在发布初始版本时，就应该将 OTA token 设置为自己的 token 值。

使用 AT+WEBSERVER 进行 OTA 升级

请参考 `AT+WEBSERVER` 和 `Web Server AT` 示例 获取更多信息。

6.15 如何实现 Bluetooth LE OTA 升级

本文档介绍 ESP-AT 中 Bluetooth LE OTA 升级，分为两部分：

- **Bluetooth LE OTA 用法**：如何通过 AT 命令和示例 APP 完成升级。

- **Bluetooth LE OTA 原理**：GATT 服务、对端操作顺序、命令包与固件包格式等协议说明，供自行实现 APP 或主机端时参考。

ESP-AT Bluetooth LE OTA 基于乐鑫 BLE OTA 方案。使用 [AT+BLEOTA](#) 初始化 Bluetooth LE OTA 后，设备会开启广播，等待对端 APP 连接，通过 Bluetooth LE 接收固件，校验通过后自动重启。

- **Bluetooth LE OTA 用法**
 - 概述
 - 准备工作
 - 使用 [ESP-AT](#) 进行 [Bluetooth LE OTA](#)
 - 示例 APP
- **Bluetooth LE OTA 原理**
 - 服务定义
 - [OTA](#) 操作顺序
 - 命令包格式
 - 固件数据包格式
 - 进度条信息
- 参考

6.15.1 Bluetooth LE OTA 用法

概述

Bluetooth LE OTA 无需 Wi-Fi 网络即可升级固件。手机或主机 APP 通过 Bluetooth LE 连接 ESP 设备，按扇区传输新固件，设备将固件写入 Flash，直至升级完成。

典型应用场景：

- 设备没有 Wi-Fi 连接，或升级时 Wi-Fi 不可用。
- 希望通过手机 APP 在本地完成固件升级。
- 除 [AT+CIUPDATE](#)、[AT+USEROTA](#)、[AT+WEBSERVER](#) 等网络 OTA 命令外，还需要基于 Bluetooth LE 的 FOTA 通道。

如需了解基于网络的 OTA，请参考[如何实现 OTA 升级](#)。

准备工作

重要：默认 AT 固件未使能 Bluetooth LE OTA。如需使用该功能，请自行[编译 ESP-AT 工程](#)，在配置工程时完成以下两项，然后重新编译并烧录固件：

1. 启用 Component config>AT>AT ble ota command support
2. 同步调整 MTU，建议设置为 512：Component config>Bluetooth>NimBLE Options > Preferred MTU size in octets

使用 ESP-AT 进行 Bluetooth LE OTA

请按以下步骤完成一次典型的 Bluetooth LE OTA 升级。完整命令说明请参考[AT+BLEOTANAME](#) 和 [AT+BLEOTA](#)。

1. （可选）设置 Bluetooth LE OTA 设备名称。最大长度为 29 字节，默认名称为 ESP-C919。
命令：

```
AT+BLEOTANAME="NAME"
```

响应：

OK

2. 初始化 Bluetooth LE OTA。初始化成功后，设备会自动开启广播。

命令：

AT+BLEOTA=1

响应：

OK

3. 在手机上打开示例 APP，连接设备并开始固件升级。详见下方示例 APP。
4. 固件传输完成并校验通过后，设备会自动重启。

备注：

- 如需设置设备名称，请在执行 AT+BLEOTA=1 前先运行 [AT+BLEOTANAME](#)。
 - AT+BLEOTA=0 用于反初始化 Bluetooth LE OTA。
 - 升级过程中请尽量让手机靠近设备，避免因 Bluetooth LE 链路较弱导致传输失败。
-

示例 APP

你可以从以下地址获取乐鑫 Bluetooth LE OTA 示例 APP：

- Android: [ESP BLE OTA Android APP](#)
- iOS: [ESP BLE OTA iOS APP](#)

安装 APP 后，需要将用于升级的固件文件（如 app.bin）手动放到 APP 对应目录下，APP 才能在升级时选择该文件：

- **Android**：将 APK 安装到手机后，先进入一次 APP（会自动创建目录），然后将固件文件放到 `Android/data/com.espressif.bleota.android/files/BLE-OTA`
- **iOS**：将 iPhone 连接到电脑，通过 Finder (macOS) 或 Apple Devices / iTunes (Windows) 打开该手机，进入 文件 > ESP-BLEOTA（或 esp-ble-ota），将固件文件拖入该 App 目录

完成上述准备后：

1. 打开手机蓝牙。
2. 打开 APP，扫描附近设备。
3. 连接到执行 AT+BLEOTA=1 后正在广播的设备。
4. 选择固件文件并开始升级。
5. 等待传输完成。设备校验固件后会自动重启。

若你需要自行实现手机 APP 或主机端，请继续阅读下方 [Bluetooth LE OTA 原理](#)。

6.15.2 Bluetooth LE OTA 原理

本节说明 Bluetooth LE OTA 的 GATT 服务、对端操作顺序以及数据包格式，供自行实现对端 APP 时参考。

服务定义

Bluetooth LE OTA Profile 包含两个服务：

- **DIS Service** (UUID 0x180A)：显示软件和硬件版本信息。
- **OTA Service** (UUID 0x8018)：用于 OTA 升级，包含 4 个特征值，如下表所示。

特征值	UUID	属性	说明
RECV_FW_CHAR	0x8020	Write, Notify	接收固件数据，并回复 ACK
PROGRESS_BAR_CHAR	0x8021	Read, Notify	读取或上报升级进度
COMMAND_CHAR	0x8022	Write, Notify	发送 OTA 命令，并回复 ACK
CUSTOMER_CHAR	0x8023	Write, Notify	收发用户自定义数据

开始 OTA 前，对端设备必须：

1. 使能所有通过 ACK 回复的特征值的 Indication（向 CCC 写入 0x0002）。
2. 发送“开始 OTA”命令，并在命令 Payload 中携带整个固件的长度。

设备每累计接收 4 KB 数据写一次 Flash。最后不足 4 KB 的扇区，在完整固件接收并校验通过后结束传输。

OTA 操作顺序

手机或主机 APP 连接设备后，请按以下顺序操作。先使能全部 Indication (CCC)，再发送开始 OTA 命令。否则设备不会接收固件数据。

步骤	对端设备操作	说明
1	使能所有通过 ACK 回复的特征值的 Indication	向 RECV_FW_CHAR、COMMAND_CHAR、PROGRESS_BAR_CHAR、CUSTOMER_CHAR 的 CCC 写入 0x02 0x00（小端，即 0x0002），以开启 Indication。按 BLE CCC 定义，0x0001 使能 Notification，0x0002 使能 Indication。设备通过 Indication 回复 ACK。
2	下发“开始 OTA”命令	向 COMMAND_CHAR 写入命令包。Command ID 0x0001 表示开始 OTA。Payload 的 Byte 2~5 携带固件总长度。等待设备通过 COMMAND_CHAR Indication 回复 0x0003，且 Byte 4~5 为 0x0000（接受）后，再进入下一步。
3	按扇区发送固件数据	使用 Write Without Response 向 RECV_FW_CHAR 写入固件包。每个扇区固定 4 KB。Sector_Index 从 0 递增且不可跳跃。每个扇区发送完成后，需等待成功 ACK，再发送下一扇区。
4	（可选）下发“结束 OTA”命令	固件数据全部发送完成后，可向 COMMAND_CHAR 写入 Command ID 0x0002 以结束 OTA。

简要流程：

连接 -> 使能 Indication -> 开始 OTA（携带固件长度）-> 按扇区发送固件（每扇区等待 ACK）-> （可选）结束 OTA -> 设备校验固件并重启

命令包格式

COMMAND_CHAR 使用如下包格式：

字段	Command_ID	PayLoad	CRC16
字节	Byte 0 ~ 1	Byte 2 ~ 17	Byte 18 ~ 19

Command_ID 定义如下：

Command_ID	含义	Payload 说明
0x0001	开始 OTA	Byte 2~5: 固件长度 (小端)。其余字节置 0。CRC16 对 Byte 0~17 计算。
0x0002	结束 OTA	Payload 全部置 0。CRC16 对 Byte 0~17 计算。
0x0003	命令包回复	Byte 2~3: 被回复的 Command_ID。Byte 4~5: 0x0000 表示接受, 0x0001 表示拒绝。其余字节置 0。

固件数据包格式

客户端按如下格式向 RECV_FW_CHAR 发送固件包:

字段	Sector_Index	Packet_Seq	PayLoad
字节	Byte 0 ~ 1	Byte 2	Byte 3 ~ (MTU_size - 4)

- **Sector_Index**: 扇区号, 从 0 递增, 不可跳跃。必须发满当前扇区的 4 KB 后才能开始下一扇区, 否则设备会立即回复错误 ACK 并要求重传。
- **Packet_Seq**: 包序号。0xFF 表示该扇区的最后一包。此时 Payload 最后 2 字节为整个 4 KB 扇区的 CRC16。
- **PayLoad**: 固件有效载荷。

回复包与命令包同为 20 字节, 未使用字节置 0。CRC16 对 Byte 0~17 计算。

字段	Sector_Index	ACK_Status	Ex-pected_Sector_In	Reserved	CRC16
字节	Byte 0 ~ 1	Byte 2 ~ 3	Byte 4 ~ 5	Byte 6 ~ 17	Byte 18 ~ 19

ACK_Status 定义如下:

- 0x0000: 成功
- 0x0001: CRC 错误
- 0x0002: Sector_Index 错误; Byte 4~5 (Expected_Sector_Index) 表示期望的 Sector_Index
- 0x0003: Payload 长度错误

进度条信息

若传输中断或包序号错乱, 客户端可主动读取 PROGRESS_BAR_CHAR, 并从期望偏移继续传输。

字节	含义
Byte 0 ~ 3	recv_total_length
Byte 4 ~ 7	total_fw_length
Byte 8 ~ 9	CRC16

6.15.3 参考

- Bluetooth LE OTA AT 命令: [AT+BLEOTANAME](#) 和 [AT+BLEOTA](#)
- 网络 OTA 指南: [如何实现 OTA 升级](#)

6.16 如何更新 ESP-IDF 版本

ESP-AT 固件基于乐鑫物联网开发框架 (ESP-IDF)，每个版本的 ESP-AT 固件对应某个特定的 ESP-IDF 版本。强烈建议使用 ESP-AT 工程默认的 ESP-IDF 版本，**不建议**更新 ESP-IDF 版本，因为 libesp32c6_at_core.a 底层的 ESP-IDF 版本与 ESP-AT 工程的 IDF 版本不一致可能会导致固件的错误操作。

但是，在某些特殊情况下，ESP-IDF 的小版本更新也可能适用于 ESP-AT 工程。如果你需要更新，本文档可作为参考。

ESP-AT 固件对应的 ESP-IDF 版本记录在 IDF_VERSION 文件中，这些文件分布在 `module_config` 文件夹下的不同模组文件夹中。该文件描述了模组固件所基于的 ESP-IDF 的分支、提交 ID 和仓库地址。例如，PLATFORM_ESP32C6 平台的 ESP32C6-4MB 模组的 IDF_VERSION 位于 `module_config/module_esp32c6_default/IDF_VERSION`。

如果你想为 ESP-AT 固件更新 ESP-IDF 版本，请按照以下步骤操作。

- 找到模组的 IDF_VERSION 文件。
- 根据需要更新其中的分支、提交 ID 和仓库地址。
- 删除 esp-at 根目录下原有的 esp-idf，以便下次编译时首先克隆 IDF_VERSION 中指定的 ESP-IDF 版本。
- 重新编译 ESP-AT 工程。

注意，当 ESP-AT 版本和 ESP-IDF 版本不匹配，编译时会报如下错误。

```
Please wait for the update to complete, which will take some time
```

6.17 ESP-AT 固件差异

文档已移至 [AT 发布固件差异](#)。

6.18 如何从 GitHub 下载最新临时版本 AT 固件

由于 ESP-AT 在 GitHub 上启用 CI（持续集成），因此每次代码被推送到 GitHub 都会生成 ESP-AT 固件的临时版本。

注意：你所下载的最新临时版本 AT 固件仅用于测试目的。若需用于量产，请务必依据你产品的实际应用场景进行全面的验证与测试。此外，当前固件不支持官方的 OTA 功能 ([AT+CIUPDATE](#))。

请保存好下载的固件以及下载链接，用于后续可能的问题调试。

以下步骤指导你如何从 GitHub 下载最新临时版本 AT 固件。

1. 登录你的 GitHub 账号
在开始之前，请先登录你的 GitHub 账号，因为下载固件需要登录权限。
2. 打开网页 <https://github.com/espressif/esp-at>
3. 点击 “Actions” 进入 “Actions” 页面
4. 点击 “Branch” 选择指定分支
默认为 master 分支。如果你想下载指定分支的临时固件，点击 “Branch” 进入指定分支的 workflow 页面。
5. 点击最新的 workflow 进入 workflow 页面
6. 将页面滚动到最后，在 Artifacts 页面中选择相对应的模组

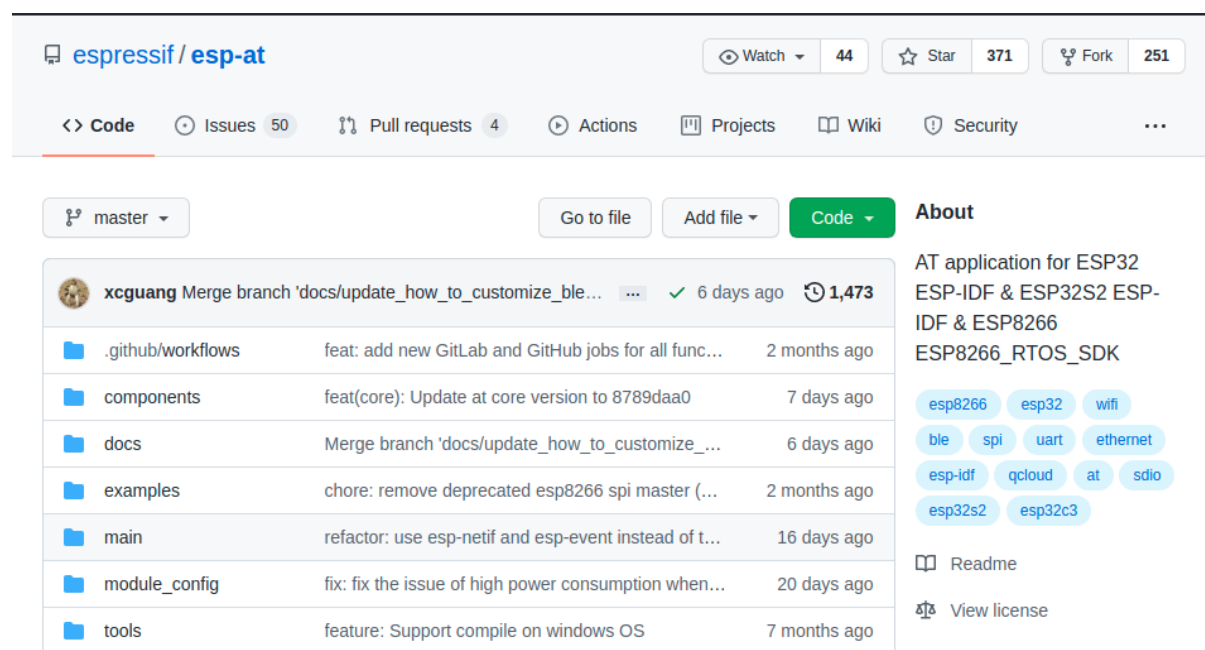


图 30: ESP-AT GitHub 官方页面

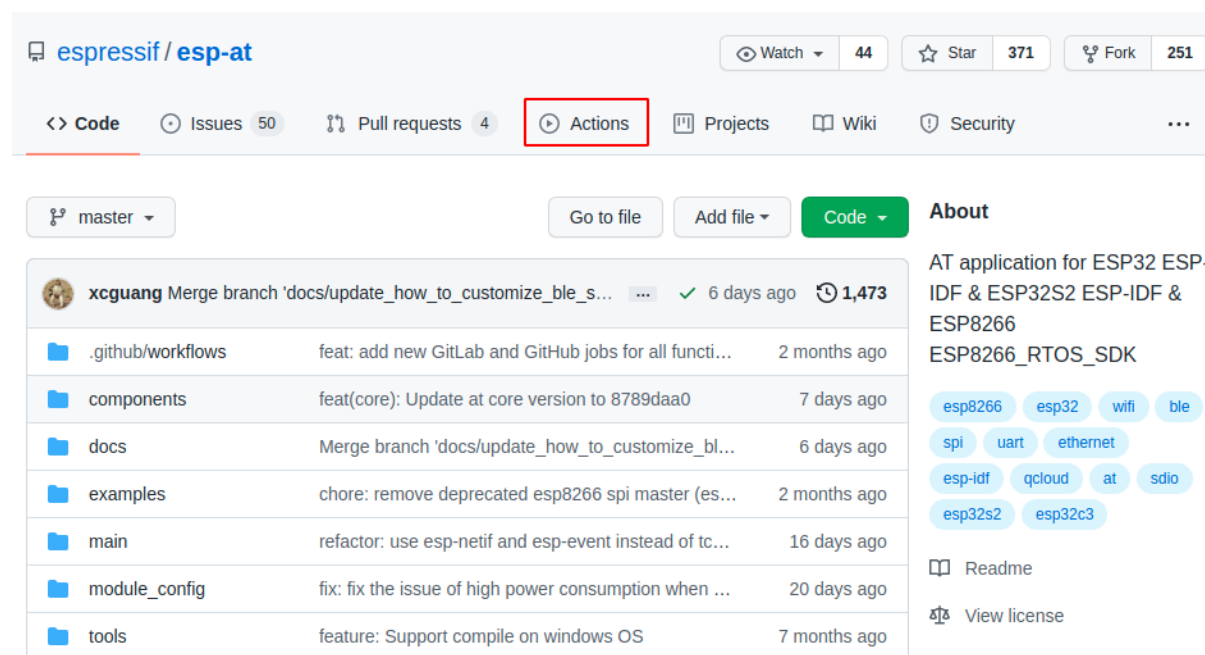


图 31: 点击 Actions

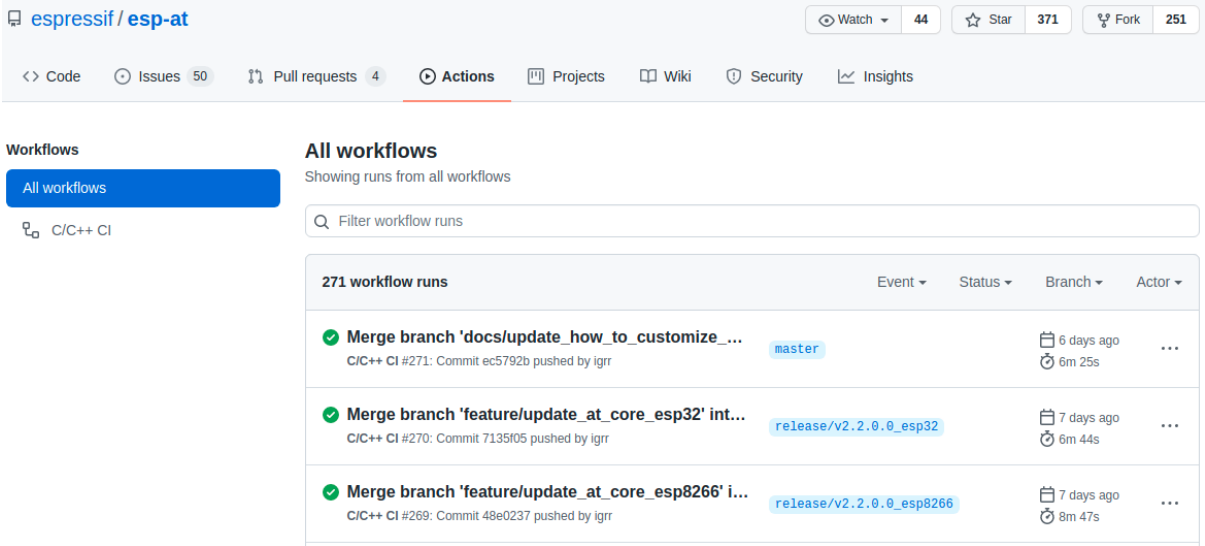


图 32: Actions 页面

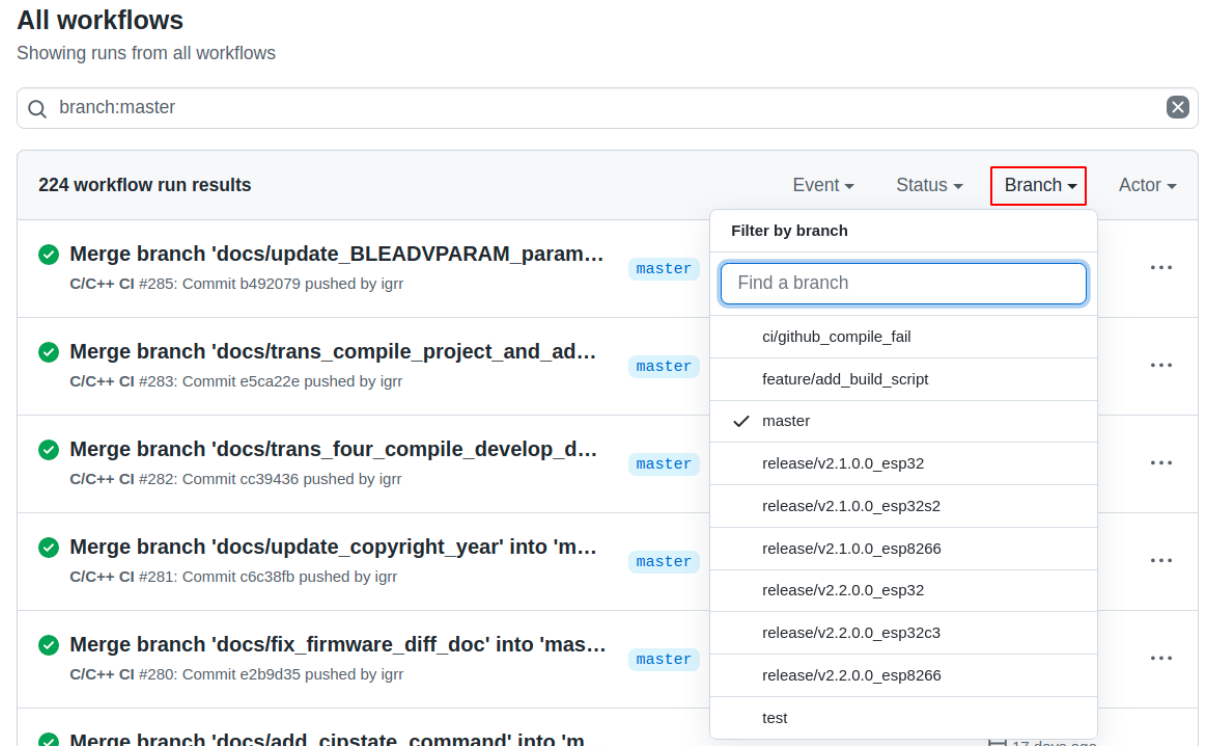


图 33: 点击 Branch

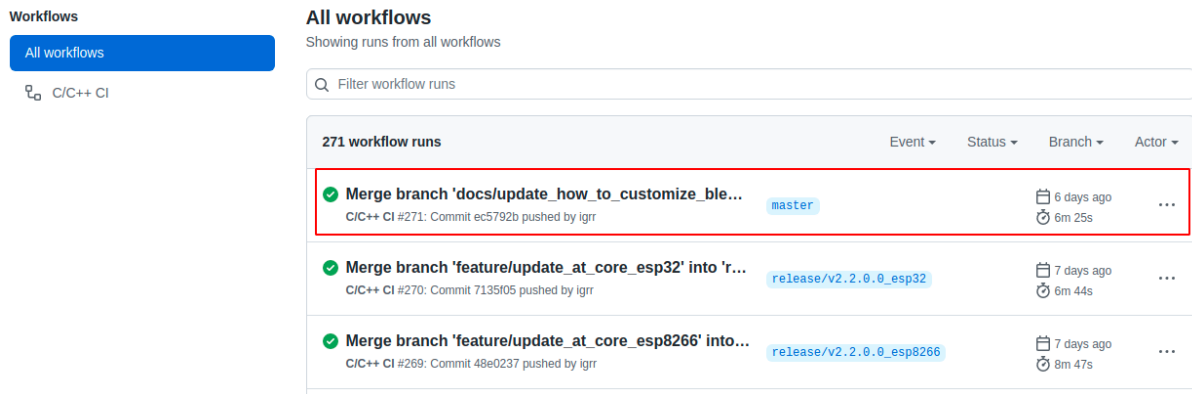


图 34: 点击最新的 Workflow

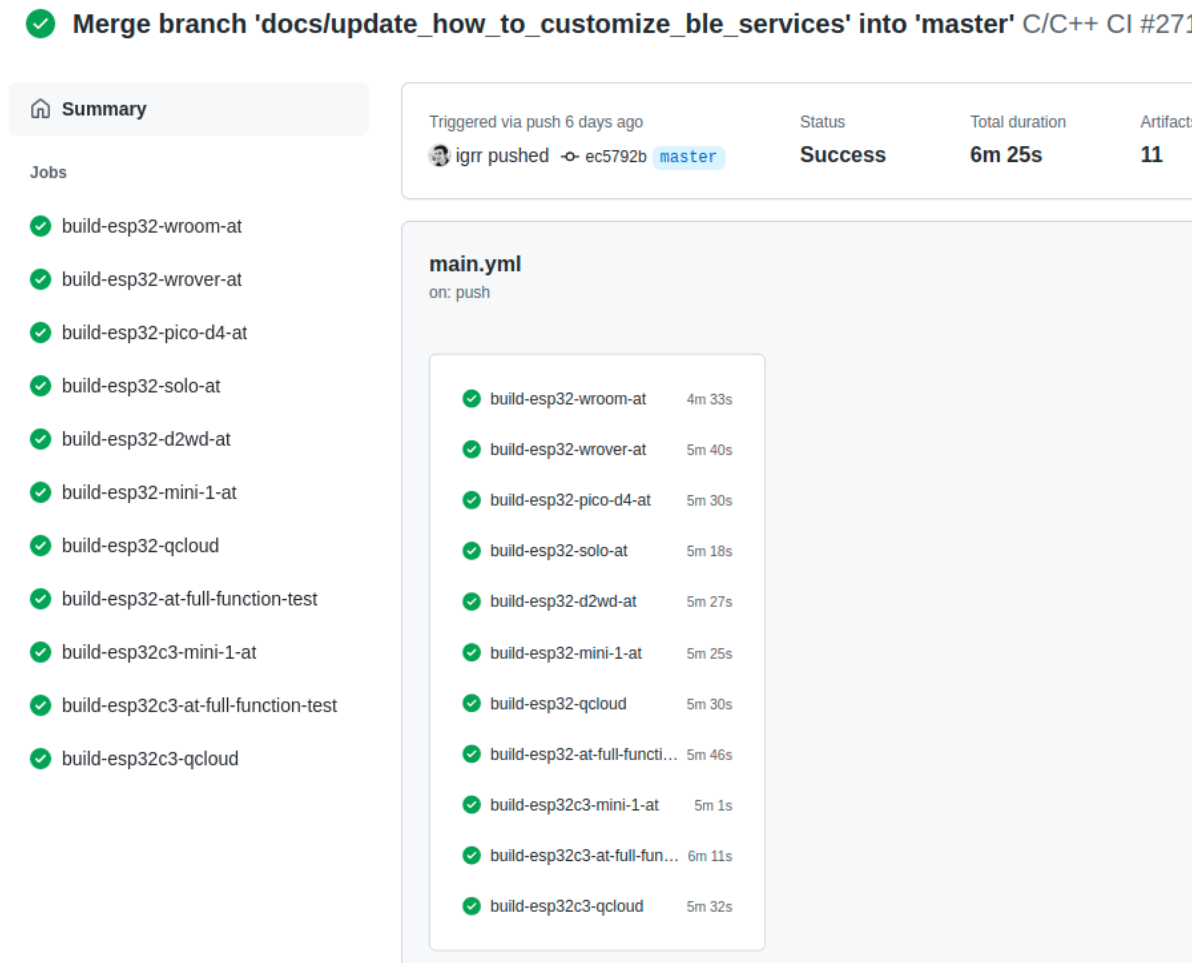


图 35: 最新的 Workflow 页面

Artifacts		
Produced during runtime		
Name		Size
 esp32-at-full-function-test		35.8 MB
 esp32-d2wd-at		26.2 MB
 esp32-mini-1-at		28.3 MB
 esp32-pico-d4-at		28.3 MB
 esp32-qcloud		29.2 MB
 esp32-solo-1-at		28.3 MB
 esp32-wroom-at		28.3 MB
 esp32-wrover-at		30.2 MB
 esp32c3-at-full-function-test		29.7 MB
 esp32c3-mini-1-at		27.5 MB
 esp32c3-qcloud		28.5 MB

图 36: Artifacts 页面

7. 点击下载模组的最新临时版本 AT 固件

备注：如果你在 Artifacts 页面中没有找到对应的模组，请参考[ESP-AT 固件差异](#) 选择类似固件进行下载。

6.19 at.py 工具

at.py 工具用于修改打包好的 2 MB/4 MB AT 量产固件（即 build/factory 目录下的 factory_XXX.bin）中的多种参数配置。这些配置包括 Wi-Fi 配置、证书和密钥配置、UART 配置、GATTS 配置等。当默认的固件无法满足你的需求时，你可以使用 at.py 工具修改固件中的这些参数配置。

6.19.1 详细步骤

请根据下方详细步骤，完成 at.py 工具下载、固件中的配置修改以及固件烧录。**推荐你优先选择 Linux 系统开发。**

- 第一步：Python 安装
- 第二步：at.py 下载
- 第三步：at.py 用法说明
- 第四步：at.py 修改固件中的配置示例
- 第五步：固件烧录

6.19.2 第一步：Python 安装

如果你本地已经安装过 Python，则可以跳过此步骤。否则，请根据 [Python 官方文档](#)，完成 Python 的下载和安装。**推荐你使用 Python 3.10.0 或更高版本。**

安装完成后，你可以在命令行中输入 `python --version`，查看 Python 版本。

6.19.3 第二步：at.py 下载

访问 [at.py](#) 页面，点击 Download raw file 按钮，将 at.py 文件下载到本地。

6.19.4 第三步：at.py 用法说明

当前 at.py 支持修改固件中的参数配置，请在命令行中输入 `python at.py modify_bin --help`，查看支持的用法以及说明。

6.19.5 第四步：at.py 修改固件中的配置示例

- 修改 Wi-Fi 配置
- 修改证书和密钥配置
- 修改 UART 配置
- 修改 GATTS 配置

备注：

- 你可以混合修改多种参数配置，例如，你可以同时修改 Wi-Fi 配置和证书和密钥配置。
- 待修改的参数配置，脚本并不会检查其合法性，请确保你输入的参数配置是合法的。

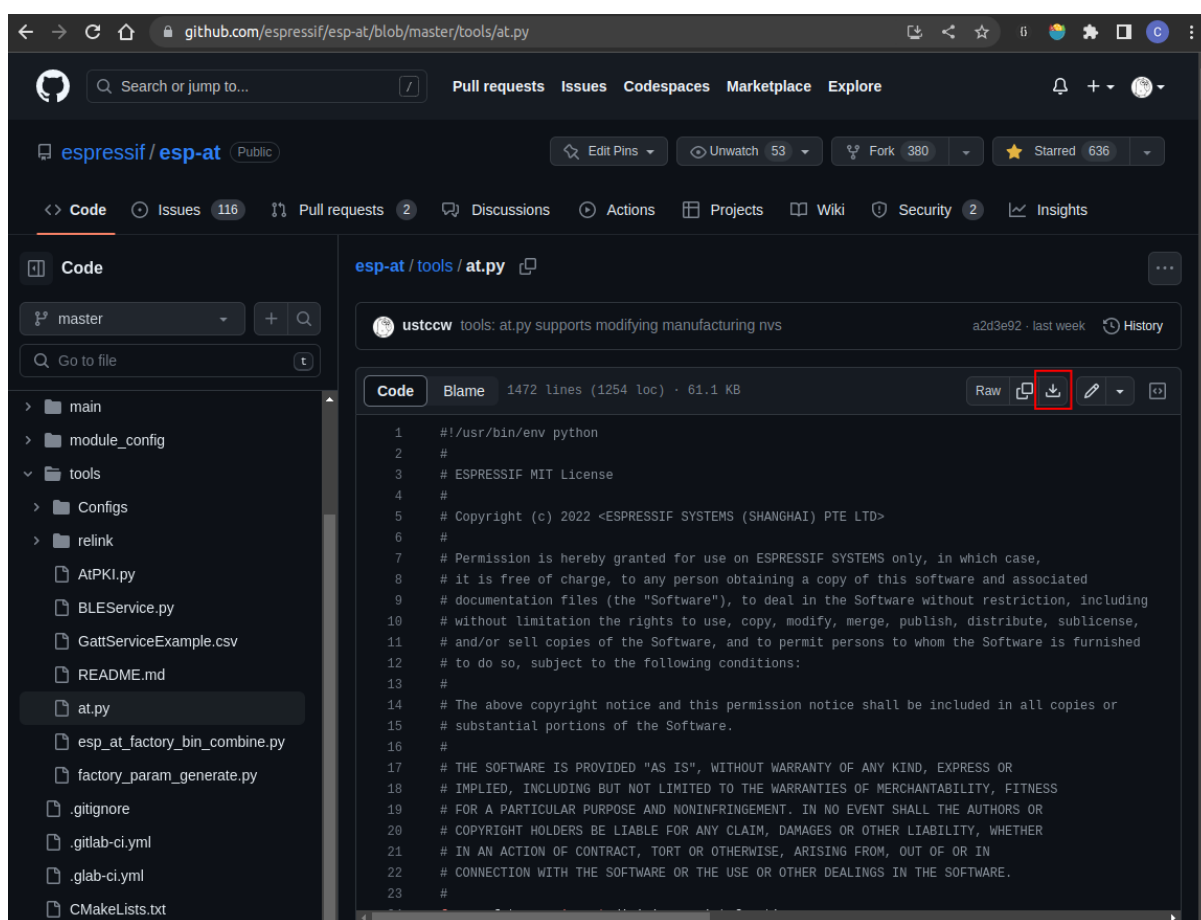


图 37: 下载 at.py 文件

- 修改后的参数配置，在当前 `mfg_nvs` 目录下的 `mfg_nvs.csv` 中，会有对应的记录。

修改 Wi-Fi 配置

当前可修改的 Wi-Fi 参数配置如下表所示：

参数	功能	说明
<code>--tx_power</code>	Wi-Fi 的最大发射功率	详情见 ESP32-C6 发射功率
<code>--country_code</code>	Wi-Fi 国家代码	详情见 Wi-Fi 国家/地区代码 中的 <code>cc</code> 字段
<code>--start_channel</code>	Wi-Fi 起始信道	详情见 Wi-Fi 国家/地区代码 中的 <code>schan</code> 字段
<code>--channel_number</code>	Wi-Fi 的总信道数	详情见 Wi-Fi 国家/地区代码 中的 <code>nchan</code> 字段

例如，你可以使用以下命令，修改 Wi-Fi 的最大发射功率为 18 dBm，国家代码为 US，起始信道为 1，总信道数为 11：

```
python at.py modify_bin --tx_power 72 --country_code "US" --start_channel 1 --  
↪channel_number 11 --input factory_XXX.bin
```

- `--tx_power 72`：单位是 0.25 dBm，72 表示 18 dBm
- `--input factory_XXX.bin`：输入的固件文件

修改证书和密钥配置

当前可修改的证书和密钥配置如下表所示：

参数	功能	原始文件
--server_ca	TLS 服务器的 CA 证书	server_ca.crt
--server_cert	TLS 服务器的证书	server_cert.crt
--server_key	TLS 服务器的密钥	server.key
--client_ca0	第 0 套客户端的 CA 证书	client_ca_00.crt
--client_cert0	第 0 套客户端的证书	client_cert_00.crt
--client_key0	第 0 套客户端的密钥	client_key_00.key
--client_ca1	第 1 套客户端的 CA 证书	client_ca_01.crt
--client_cert1	第 1 套客户端的证书	client_cert_01.crt
--client_key1	第 1 套客户端的密钥	client_key_01.key
--mqtt_ca	MQTT 客户端的 CA 证书	mqtt_ca.crt
--mqtt_cert	MQTT 客户端的证书	mqtt_client.crt
--mqtt_key	MQTT 客户端的密钥	mqtt_client.key
--wpa_ent_ca	Wi-Fi 企业版客户端 (WPA2-Enterprise/WPA3-Enterprise) 的 CA 证书	wpa_ent_ca.pem
--wpa_ent_cert	Wi-Fi 企业版客户端 (WPA2-Enterprise/WPA3-Enterprise) 的证书	wpa_ent_client.crt
--wpa_ent_key	Wi-Fi 企业版客户端 (WPA2-Enterprise/WPA3-Enterprise) 的密钥	wpa_ent_client.key
--wpa2_ca	[legacy] WPA2-Enterprise 客户端的 CA 证书，仅当固件使能 CONFIG_AT_EAP_LEGACY_NAME 时有效	wpa_ent_ca.pem
--wpa2_cert	[legacy] WPA2-Enterprise 客户端的证书，仅当固件使能 CONFIG_AT_EAP_LEGACY_NAME 时有效	wpa_ent_client.crt
--wpa2_key	[legacy] WPA2-Enterprise 客户端的密钥，仅当固件使能 CONFIG_AT_EAP_LEGACY_NAME 时有效	wpa_ent_client.key

例如，你可以使用以下命令，修改 MQTT 客户端的 CA 证书、客户端的证书和密钥。

```
python at.py modify_bin --mqtt_ca mqtt/mqtt_ca.crt --mqtt_cert mqtt/mqtt.crt --  
↪ mqtt_key mqtt/mqtt.key --input factory_XXX.bin
```

- **--input factory_XXX.bin**: 输入的固件文件

修改 UART 配置

可修改的 UART 配置，仅包括 [AT 命令端口](#) 的 UART 配置。可修改的参数配置如下表所示：

参数	功能	说明
--uart_num	AT 命令口的 UART 号	仅在 AT 日志口同时用作 AT 命令口时，需要修改此参数。同时需保证下面的 tx_pin 和 rx_pin 要配置与 AT 日志端口的 tx 和 rx 管脚一致，如果 AT 日志端口只配置了 rx，则下面 tx_pin 需要配置与下载固件口（可查看 硬件连接 ）的 UART 的 tx 管脚一致。
--baud	AT 命令口的波特率	原始值：115200。
--tx_pin	AT 命令口的 TX 管脚	请保证待修改的管脚不会和其他管脚冲突。
--rx_pin	AT 命令口的 RX 管脚	请保证待修改的管脚不会和其他管脚冲突。
--cts_pin	AT 命令口的 CTS 管脚	请保证待修改的管脚不会和其他管脚冲突。不用流控时，修改此参数为 -1。
--rts_pin	AT 命令口的 RTS 管脚	请保证待修改的管脚不会和其他管脚冲突。不用流控时，修改此参数为 -1。

例如，你可以使用以下命令，修改 AT 命令口的波特率为 921600，TX 管脚为 17，RX 管脚为 16，禁用流控。

```
python at.py modify_bin --baud 921600 --tx_pin 17 --rx_pin 16 --cts_pin -1 --rts_pin -1 --input factory_XXX.bin
```

- --input factory_XXX.bin：输入的固件文件

修改 GATTS 配置

修改前，请先阅读[如何自定义低功耗蓝牙服务](#)文档，了解 GATTS 的配置文件 [gatts_data.csv](#) 中的各个字段的含义。

当前可修改的 GATTS 配置如下表所示：

参数	功能
--gatts_cfg0	更新 gatts_data.csv 文件中 index 为 0 的一行数据
--gatts_cfg1	更新 gatts_data.csv 文件中 index 为 1 的一行数据
--gatts_cfg2	更新 gatts_data.csv 文件中 index 为 2 的一行数据
--gatts_cfg3	更新 gatts_data.csv 文件中 index 为 3 的一行数据
--gatts_cfg4	更新 gatts_data.csv 文件中 index 为 4 的一行数据
--gatts_cfg5	更新 gatts_data.csv 文件中 index 为 5 的一行数据
--gatts_cfg6	更新 gatts_data.csv 文件中 index 为 6 的一行数据
--gatts_cfg7	更新 gatts_data.csv 文件中 index 为 7 的一行数据
--gatts_cfg8	更新 gatts_data.csv 文件中 index 为 8 的一行数据
--gatts_cfg9	更新 gatts_data.csv 文件中 index 为 9 的一行数据
--gatts_cfg10	更新 gatts_data.csv 文件中 index 为 10 的一行数据
--gatts_cfg11	更新 gatts_data.csv 文件中 index 为 11 的一行数据
--gatts_cfg12	更新 gatts_data.csv 文件中 index 为 12 的一行数据
--gatts_cfg13	更新 gatts_data.csv 文件中 index 为 13 的一行数据
--gatts_cfg14	更新 gatts_data.csv 文件中 index 为 14 的一行数据
--gatts_cfg15	更新 gatts_data.csv 文件中 index 为 15 的一行数据
--gatts_cfg16	更新 gatts_data.csv 文件中 index 为 16 的一行数据
--gatts_cfg17	更新 gatts_data.csv 文件中 index 为 17 的一行数据
--gatts_cfg18	更新 gatts_data.csv 文件中 index 为 18 的一行数据
--gatts_cfg19	更新 gatts_data.csv 文件中 index 为 19 的一行数据
--gatts_cfg20	更新 gatts_data.csv 文件中 index 为 20 的一行数据
--gatts_cfg21	更新 gatts_data.csv 文件中 index 为 21 的一行数据
--gatts_cfg22	更新 gatts_data.csv 文件中 index 为 22 的一行数据

续下页

表 11 – 接上页

参数	功能
--gatts_cfg23	更新 gatts_data.csv 文件中 index 为 23 的一行数据
--gatts_cfg24	更新 gatts_data.csv 文件中 index 为 24 的一行数据
--gatts_cfg25	更新 gatts_data.csv 文件中 index 为 25 的一行数据
--gatts_cfg26	更新 gatts_data.csv 文件中 index 为 26 的一行数据
--gatts_cfg27	更新 gatts_data.csv 文件中 index 为 27 的一行数据
--gatts_cfg28	更新 gatts_data.csv 文件中 index 为 28 的一行数据
--gatts_cfg29	更新 gatts_data.csv 文件中 index 为 29 的一行数据
--gatts_cfg30	更新 gatts_data.csv 文件中 index 为 30 的一行数据

例如，你可以使用以下命令，修改 index 为 0 行的 perm 权限。

```
python at.py modify_bin --gatts_cfg0 "0,16,0x2800,0x011,2,2,A002" --input factory_
↪XXX.bin
```

- **--input factory_XXX.bin**: 输入的固件文件

注意：修改后的 AT 固件，需要你根据自己的产品自行测试验证功能。
请保存好修改前和修改后的固件以及下载链接，用于后续可能的问题调试。

第五步：固件烧录 请根据[固件烧录指南](#)，完成固件烧录。

6.20 AT API Reference

6.20.1 Header File

- [components/at/include/esp_at_core.h](#)

6.20.2 Functions

bool **esp_at_custom_cmd_array_register** (const [esp_at_cmd_t](#) *custom_at_cmd_array, uint32_t cmd_num)

Register a custom AT command set.

参数

- **custom_at_cmd_array** –Array of [esp_at_cmd_t](#) descriptors.
- **cmd_num** –Number of entries in the array.

返回

true on success, false if registration failed.

void **esp_at_device_ops_register** ([esp_at_intf_ops_t](#) *ops)

Register the device I/O operations for the AT transport interface.

参数

ops –Pointer to a populated [esp_at_intf_ops_t](#).

bool **esp_at_custom_net_ops_register** (int32_t link_id, [esp_at_net_ops_t](#) *ops)

Register passthrough callbacks for a socket-based network link.

备注： Call this only after `esp_at_module_init()`.

参数

- **link_id** –Link identifier (0-based).
- **ops** –Pointer to a populated *esp_at_net_ops_t*.

返回

true on success.

bool **esp_at_custom_ble_ops_register** (int32_t conn_index, *esp_at_ble_ops_t* *ops)

Register passthrough callbacks for a BLE link.

备注: Call this only after esp_at_module_init().

参数

- **conn_index** –BLE connection index (0-based).
- **ops** –Pointer to a populated *esp_at_ble_ops_t*.

返回

true on success.

void **esp_at_custom_ops_register** (*esp_at_custom_ops_t* *ops)

Register lifecycle callbacks for AT core events.

参数

ops –Pointer to a populated *esp_at_custom_ops_t*.

const uint8_t ***esp_at_get_current_cmd_name** (void)

Return the name of the AT command currently being executed.

返回

Pointer to a NUL-terminated command name string. Valid only during command handler execution.

const esp_partition_t ***esp_at_custom_partition_find** (esp_partition_type_t type,
esp_partition_subtype_t subtype, const char
*label)

Look up a flash partition in the secondary (user) partition table.

Searches only partitions defined in at_customize.csv (the secondary partition table), not the primary system partition table partitions_at.csv.

备注: To look up the primary partition table first and fall back to the secondary table when no match is found, use esp_partition_find_first(). In ESP-AT, this IDF API is linker-wrapped to perform that two-stage lookup.

参数

- **type** –Partition type.
- **subtype** –Partition subtype.
- **label** –Partition label string.

返回

Pointer to the matching esp_partition_t, or NULL if not found.

esp_at_para_parse_ret_t **esp_at_get_para_as_digit** (int32_t para_index, int32_t *value)

Parse the parameter at para_index as a signed 32-bit integer.

参数

- **para_index** –Zero-based parameter position in the AT command line.
- **value** –Output: parsed integer value.

返回

ESP_AT_PARA_PARSE_RET_OK, _FAIL, or _OMITTED.

esp_at_para_parse_ret_t **esp_at_get_para_as_float** (int32_t para_index, float *value)

Parse the parameter at para_index as a single-precision float.

参数

- **para_index** –Zero-based parameter position.
- **value** –Output: parsed float value.

返回

ESP_AT_PARA_PARSE_RET_OK, _FAIL, or _OMITTED.

esp_at_para_parse_ret_t **esp_at_get_para_as_str** (int32_t para_index, uint8_t **result)

Return a pointer to the raw string for the parameter at para_index.

The returned pointer refers to the AT core's internal command buffer. The caller must not free or modify the pointed-to string.

参数

- **para_index** –Zero-based parameter position.
- **result** –Output: pointer to the NUL-terminated parameter string.

返回

ESP_AT_PARA_PARSE_RET_OK, _FAIL, or _OMITTED.

bool **esp_at_port_recv_data_notify** (int32_t len, uint32_t msec)

Notify the AT core that len bytes have arrived on the port (task context).

参数

- **len** –Number of newly received bytes.
- **msec** –Maximum time to wait for the core to accept the notification (ms).

返回

true if the notification was accepted within the timeout.

int32_t **esp_at_port_read_data** (uint8_t *data, int32_t len)

Read up to len bytes from the AT input port.

参数

- **data** –Destination buffer.
- **len** –Maximum number of bytes to read.

返回

Number of bytes actually read, or negative on error.

int32_t **esp_at_port_write_data** (uint8_t *data, int32_t len)

Write len bytes to the AT output port.

The output is subject to AT+SYMSGFILTER (when filtering is enabled). The host MCU is not woken before the data is sent.

参数

- **data** –Data buffer.
- **len** –Number of bytes to write.

返回

Number of bytes written, or negative on error.

int32_t **esp_at_port_active_write_data** (uint8_t *data, int32_t len)

Write len bytes to the AT output port, waking the host MCU first.

The output is subject to AT+SYMSGFILTER (when filtering is enabled). Before sending, the pre-write callback registered via *esp_at_custom_ops_register()* is invoked, which wakes the host MCU according to the AT+USERWKMUCCFG configuration.

参数

- **data** –Data buffer.
- **len** –Number of bytes to write.

返回

Number of bytes written, or negative on error.

`int32_t esp_at_port_write_data_without_filter (uint8_t *data, int32_t len)`

Write `len` bytes to the AT output port, bypassing AT+SYMSGFILTER.

The output is never filtered by AT+SYMSGFILTER. The host MCU is not woken before the data is sent.

参数

- **data** –Data buffer.
- **len** –Number of bytes to write.

返回

Number of bytes written, or negative on error.

`int32_t esp_at_port_active_write_data_without_filter (uint8_t *data, int32_t len)`

Write `len` bytes to the AT output port, waking the host MCU first and bypassing AT+SYMSGFILTER.

The output is never filtered by AT+SYMSGFILTER. Before sending, the pre-write callback registered via [esp_at_custom_ops_register\(\)](#) is invoked, which wakes the host MCU according to the AT+USERWKMCMCFG configuration.

参数

- **data** –Data buffer.
- **len** –Number of bytes to write.

返回

Number of bytes written, or negative on error.

`bool esp_at_port_wait_write_complete (int32_t timeout_msec)`

Block until the AT output port transmit buffer is empty.

参数

timeout_msec –Maximum wait time in milliseconds.

返回

true if the buffer drained before the timeout.

`int32_t esp_at_port_get_data_length (void)`

Return the number of bytes currently pending on the AT input port.

返回

- `>= 0` : the length of the data received
- others: failure

`void esp_at_port_enter_specific (esp_at_port_specific_callback_t callback)`

Enter a special receive mode; `callback` is invoked on each data arrival.

参数

callback –Function called when new data arrives.

`void esp_at_port_exit_specific (void)`

Exit the special receive mode entered by [esp_at_port_enter_specific\(\)](#).

`void esp_at_write_result (uint8_t result_code)`

Output a standard result string (OK, ERROR, SEND OK, etc.) on the AT port.

This only writes the corresponding result string to the AT port; it does not change the state of the AT port receive task. Use it when, after emitting the result string, the command handler still needs to run further logic that is unrelated to receiving new data on the AT port.

备注: If the command is finished and the AT port should resume accepting new input, use [esp_at_dispatch_result\(\)](#) instead.

参数

result_code –One of the `esp_at_rc_t` values cast to `uint8_t`.

void **esp_at_dispatch_result** (*esp_at_rc_t* code, void *pbuf)

Output a result string and return the AT port to the ready state.

In addition to outputting the corresponding result string (like *esp_at_write_result()*), this dispatches the result code through any registered result handlers and brings the AT command receive task back to its idle/ready state, so that the AT port can accept and process new input.

Use it when the command handler has finished and the next step is to receive fresh data from the AT port. For example, after printing “OK” and the “>” data prompt, call this so the AT port is ready to receive the new input that the handler expects.

参数

- **code** –Result code to dispatch.
- **pbuf** –Optional command-specific data; set it to NULL if not needed.

void **esp_at_restart** (void)

Restart the system with a hardware watchdog safety net.

Arms a 3-second hardware timer before calling the *esp_restart()* routine. If the normal restart path stalls, the watchdog forces a hard reset. No output is generated on the AT port before the restart.

void **esp_at_restart_async** (void)

Respond with OK on the AT port, run the *pre_restart* callback, then restart.

Use this variant to restart in response to an AT command so that the host receives an OK acknowledgement before the device reboots. Arms the same 3-second hardware watchdog as *esp_at_restart()*.

bool **esp_at_str_2_mac** (const char *str, uint8_t mac[6])

Convert a colon-separated MAC address string to a 6-byte binary array.

Accepts the format “XX:XX:XX:XX:XX:XX” (case-insensitive hex digits).

参数

- **str** –NUL-terminated MAC address string.
- **mac** –Output: 6-byte binary MAC address.

返回

true on success, false if the string is malformed.

6.20.3 Structures

struct **esp_at_cmd_t**

Descriptor for a single AT command.

Register an array of these via *esp_at_custom_cmd_array_register()*. Set unused command-type handlers to NULL.

Public Members

char ***cmd_name**

Command name string, e.g. “+MYCOMMAND”

uint8_t (***test_cmd**)(uint8_t *cmd_name)

Handler for AT+CMD=?

uint8_t (***query_cmd**)(uint8_t *cmd_name)

Handler for AT+CMD?

uint8_t (***setup_cmd**)(uint8_t para_num)

Handler for AT+CMD=<params>

uint8_t (***exe_cmd**)(uint8_t *cmd_name)

Handler for AT+CMD

struct **esp_at_intf_ops_t**

I/O operation callbacks for the underlying AT transport device.

Pass a populated instance to [*esp_at_device_ops_register\(\)*](#).

Public Members

int32_t (***read_data**)(uint8_t *data, int32_t len)

Read bytes from the physical interface

int32_t (***write_data**)(uint8_t *data, int32_t len)

Write bytes to the physical interface

int32_t (***get_data_length**)(void)

Return the number of bytes available

bool (***wait_write_complete**)(int32_t timeout_msec)

Block until the transmit buffer is drained

struct **esp_at_net_ops_t**

Callbacks for a socket-based passthrough (transparent-transmission) link.

Register via [*esp_at_custom_net_ops_register\(\)*](#).

Public Members

int32_t (***recv_data**)(uint8_t *data, int32_t len)

Invoked when data arrives on the link

void (***connect_cb**)(void)

Invoked when the link is established

void (***disconnect_cb**)(void)

Invoked when the link is torn down

struct **esp_at_ble_ops_t**

Callbacks for a BLE passthrough link.

Register via [*esp_at_custom_ble_ops_register\(\)*](#).

Public Members

`int32_t (*recv_data)(uint8_t *data, int32_t len)`

Invoked when data arrives on the BLE link

`void (*connect_cb)(void)`

Invoked when the BLE link is established

`void (*disconnect_cb)(void)`

Invoked when the BLE link is torn down

`struct esp_at_custom_ops_t`

Lifecycle and status callbacks for AT core events.

Pass a populated instance to *esp_at_custom_ops_register()*.

Public Members

`void (*status_callback)(esp_at_status_t status)`

Called on AT core status change

`void (*pre_sleep_callback)(esp_at_sleep_mode_t mode)`

Called before entering sleep

`void (*pre_wakeup_callback)(void)`

Called before waking from sleep

`void (*pre_deepsleep_callback)(void)`

Called before entering deep sleep

`void (*pre_restart_callback)(void)`

Called before system restart

`void (*pre_active_write_data_callback)(int32_t (*write_fn)(uint8_t *data, int32_t len))`

Called before an active data write

6.20.4 Macros

ESP_AT_ERROR_NO (subcategory, extension)

Encode an AT error code from its constituent fields.

参数

- **subcategory** –Error sub-category (*esp_at_errno_t*).
- **extension** –Command-specific extension value.

返回

Packed 32-bit error code: [module:8][sub:8][ext:16].

ESP_AT_CMD_ERROR_OK

No error

ESP_AT_CMD_ERROR_NON_FINISH

Command terminator (“\r\n”) not found

ESP_AT_CMD_ERROR_NOT_FOUND_AT

Command does not start with “AT”

ESP_AT_CMD_ERROR_PARA_LENGTH (which_para)

Parameter length mismatch (which_para: 0-based index)

ESP_AT_CMD_ERROR_PARA_TYPE (which_para)

Parameter type mismatch (which_para: 0-based index)

ESP_AT_CMD_ERROR_PARA_NUM (need, given)

Parameter count mismatch (need expected, given received)

ESP_AT_CMD_ERROR_PARA_INVALID (which_para)

Invalid parameter value (which_para: 0-based index)

ESP_AT_CMD_ERROR_PARA_PARSE_FAIL (which_para)

Failed to parse a parameter (which_para: 0-based index)

ESP_AT_CMD_ERROR_CMD_UNSupport

Command not supported

ESP_AT_CMD_ERROR_CMD_EXEC_FAIL (result)

Command execution failed (result: handler-specific code)

ESP_AT_CMD_ERROR_CMD_PROCESSING

A previous command is still being processed

ESP_AT_CMD_ERROR_CMD_OP_ERROR

Unsupported command operation type

6.20.5 Type Definitions

```
typedef void (*esp_at_port_specific_callback_t)(void)
```

Callback invoked while the AT core is in a specific receive mode.

6.20.6 Enumerations

```
enum esp_at_status_t
```

Operational status of the AT core.

Values:

enumerator **ESP_AT_STATUS_NORMAL**

Command mode: processing AT commands

enumerator **ESP_AT_STATUS_TRANSMIT**

Passthrough mode: forwarding raw data

enum **esp_at_sleep_mode_t**

Supported sleep mode of the AT core.

Values:

enumerator **ESP_AT_SLEEP_DISABLE**

Sleep disabled

enumerator **ESP_AT_SLEEP_MIN_MODEM**

Minimum modem sleep (DTIM-based)

enumerator **ESP_AT_SLEEP_LIGHT**

Light sleep

enumerator **ESP_AT_SLEEP_MAX_MODEM**

Maximum modem sleep

enumerator **ESP_AT_SLEEP_MODE_MAX**

Sentinel value; not a valid mode

enum **esp_at_para_parse_ret_t**

Return value of the parameter-parsing helpers.

Values:

enumerator **ESP_AT_PARA_PARSE_RET_FAIL**

Parse failed; parameter is malformed

enumerator **ESP_AT_PARA_PARSE_RET_OK**

Parameter parsed successfully

enumerator **ESP_AT_PARA_PARSE_RET_OMITTED**

Parameter was omitted (empty field)

enum **esp_at_rc_t**

Result code selectors for [*esp_at_write_result\(\)*](#) and [*esp_at_dispatch_result\(\)*](#).

Values:

enumerator **ESP_AT_RESULT_CODE_OK**

OK

enumerator **ESP_AT_RESULT_CODE_ERROR**

ERROR

enumerator **ESP_AT_RESULT_CODE_FAIL**

ERROR

enumerator **ESP_AT_RESULT_CODE_SEND_OK**

SEND OK

enumerator **ESP_AT_RESULT_CODE_SEND_FAIL**
SEND FAIL

enumerator **ESP_AT_RESULT_CODE_IGNORE**
(no output)

enumerator **ESP_AT_RESULT_CODE_PROCESS_DONE**
(processing complete, no output)

enumerator **ESP_AT_RESULT_CODE_OK_AND_INPUT_PROMPT**
OK followed by input prompt

enumerator **ESP_AT_RESULT_CODE_MAX**
Sentinel; not a valid code

enum **esp_at_module_t**
AT core module identifier, used as the high byte of error codes.
Values:

enumerator **ESP_AT_MODULE_NUM**

enum **esp_at_errno_t**
AT command error sub-categories.
Values:

enumerator **ESP_AT_SUB_OK**
No error

enumerator **ESP_AT_SUB_COMMON_ERROR**
Generic error

enumerator **ESP_AT_SUB_NO_TERMINATOR**
Missing line terminator

enumerator **ESP_AT_SUB_NO_AT**
Missing AT prefix

enumerator **ESP_AT_SUB_PARA_LENGTH_MISMATCH**
Parameter length out of range

enumerator **ESP_AT_SUB_PARA_TYPE_MISMATCH**
Parameter type mismatch

enumerator **ESP_AT_SUB_PARA_NUM_MISMATCH**
Wrong number of parameters

enumerator **ESP_AT_SUB_PARA_INVALID**
Parameter value out of valid range

enumerator **ESP_AT_SUB PARA_PARSE_FAIL**

Failed to parse parameter

enumerator **ESP_AT_SUB UNSUPPORT_CMD**

Command not supported

enumerator **ESP_AT_SUB_CMD_EXEC_FAIL**

Command execution failed

enumerator **ESP_AT_SUB_CMD_PROCESSING**

A command is already in progress

enumerator **ESP_AT_SUB_CMD_OP_ERROR**

Wrong operation type for command

6.20.7 Header File

- [components/at/include/esp_at.h](#)

6.20.8 Functions

void **esp_at_log_write** (esp_log_level_t level, const char *tag, const char *format, ...)

Write a formatted message to the AT log output (weak, overridable).

Not intended to be called directly; use the ESP_AT_LOGE/W/I/D/V macros instead. Must not be called from an interrupt context. As a weak symbol, the default implementation can be overridden to redirect AT log output.

参数

- **level** –Log severity level.
- **tag** –Log tag string.
- **format** –printf-style format string.
- ... –Arguments referenced by *format*.

const char ***esp_at_get_current_module_name** (void)

Return the name of the currently selected AT module.

返回

NUL-terminated module name string.

void **esp_at_ready_before** (void)

Hook invoked immediately before the AT stack signals ready.

Override this weak function to: a) Execute preset AT commands via `esp_at_exe_cmd()`. b) Perform additional initialization using ESP-IDF or AT APIs.

esp_err_t **esp_at_nvs_set_str** (nvs_handle_t handle, const char *key, const char *value)

Store a NUL-terminated string in NVS (weak, overridable).

The AT core uses this helper internally to persist string data to NVS. It is a weak symbol whose default implementation simply forwards to `nvs_set_str()`. Override it to hook the storage path, for example to encrypt the value before it is written to flash (override [esp_at_nvs_get_str\(\)](#) to decrypt on read).

参见:

`nvs_set_str`

参数

- **handle** –NVS handle opened with `nvs_open()`.
- **key** –Key name (NUL-terminated).
- **value** –NUL-terminated string to store.

返回

ESP_OK on success, or an NVS error code otherwise.

`esp_err_t esp_at_nvs_get_str (nvs_handle_t handle, const char *key, char *out_value, size_t *length)`

Read a NUL-terminated string from NVS (weak, overridable).

Counterpart of `esp_at_nvs_set_str()`: the AT core uses it internally to load string data from NVS. It is a weak symbol whose default implementation simply forwards to `nvs_get_str()`. Override it together with `esp_at_nvs_set_str()` to, for example, decrypt the value after reading it from flash.

参见:

`nvs_get_str`

参数

- **handle** –NVS handle opened with `nvs_open()`.
- **key** –Key name (NUL-terminated).
- **out_value** –Buffer that receives the string; pass NULL to query the required length.
- **length** –In/out: capacity of `out_value` in bytes, updated with the actual length.

返回

ESP_OK on success, or an NVS error code otherwise.

`esp_err_t esp_at_nvs_set_blob (nvs_handle_t handle, const char *key, const void *value, size_t length)`

Store a binary blob in NVS (weak, overridable).

The AT core uses this helper internally to persist binary data to NVS. It is a weak symbol whose default implementation simply forwards to `nvs_set_blob()`. Override it to hook the storage path, for example to encrypt the blob before it is written to flash (override `esp_at_nvs_get_blob()` to decrypt on read).

参见:

`nvs_set_blob`

参数

- **handle** –NVS handle opened with `nvs_open()`.
- **key** –Key name (NUL-terminated).
- **value** –Pointer to the blob to store.
- **length** –Size of `value` in bytes.

返回

ESP_OK on success, or an NVS error code otherwise.

`esp_err_t esp_at_nvs_get_blob (nvs_handle_t handle, const char *key, void *out_value, size_t *length)`

Read a binary blob from NVS (weak, overridable).

Counterpart of `esp_at_nvs_set_blob()`: the AT core uses it internally to load binary data from NVS. It is a weak symbol whose default implementation simply forwards to `nvs_get_blob()`. Override it together with `esp_at_nvs_set_blob()` to, for example, decrypt the blob after reading it from flash.

参见:

`nvs_get_blob`

参数

- **handle** –NVS handle opened with `nvs_open()`.
- **key** –Key name (NUL-terminated).
- **out_value** –Buffer that receives the blob; pass NULL to query the required length.
- **length** –In/out: capacity of `out_value` in bytes, updated with the actual length.

返回

ESP_OK on success, or an NVS error code otherwise.

void **esp_at_yield_if_idle_timeout** (uint32_t idle_timeout_ms, uint32_t yield_ticks)

Yield the current task if the FreeRTOS idle task has not run recently.

Measures the time since the last idle hook execution. If that interval exceeds `idle_timeout_ms`, the calling task delays for `yield_ticks` RTOS ticks to allow lower-priority tasks (including the idle task) to run.

Intended to prevent high-throughput tasks from starving the idle task and triggering the task watchdog timer under sustained CPU load.

备注: Must not be called from interrupt context.

参数

- **idle_timeout_ms** –Threshold (ms) since last idle hook execution.
- **yield_ticks** –RTOS ticks to delay when a yield is triggered.

const char ***esp_at_fs_get_mount_point** (void)

Return the mount point of the AT filesystem partition.

返回

NUL-terminated mount point path string (e.g. “/littlefs”).

bool **esp_at_fs_mount** (void)

Mount the AT filesystem partition.

返回

true on success, false on failure.

bool **esp_at_fs_unmount** (void)

Unmount the AT filesystem partition.

返回

true on success, false on failure.

esp_err_t **esp_at_get_fs_info** (uint32_t *out_total_bytes, uint32_t *out_free_bytes)

Query the total and free capacity of the AT filesystem.

参数

- **out_total_bytes** –Output: total filesystem size in bytes.
- **out_free_bytes** –Output: available free space in bytes.

返回

ESP_OK on success, or an error code.

esp_at_fs_type_t **esp_at_fs_get_type** (void)

Return the filesystem type in use.

返回

The active `esp_at_fs_type_t` value.

6.20.9 Macros

ESP_AT_PORT_TX_WAIT_MS_MAX

Maximum time (ms) to wait for the AT port TX buffer to drain.

ESP_AT_BUF_ON_STACK_SIZE

Default maximum stack-allocated buffer size for AT command handlers.

esp_at_min (x, y)

Returns the smaller of two values.

esp_at_max (x, y)

Returns the larger of two values.

ESP_AT_LOG_BUFFER_HEXDUMP (tag, buffer, buff_len, level)

NULL-safe wrapper around ESP_LOG_BUFFER_HEXDUMP.

Outputs a hexadecimal buffer dump only when `buffer` is non-NULL.

CONFIG_AT_LOG_DEFAULT_LEVEL

Compile-time fallback for the AT log verbosity level when not set via Kconfig.

ESP_AT_LOGE (tag, format, ...)

Log at ERROR level, gated by CONFIG_AT_LOG_DEFAULT_LEVEL.

ESP_AT_LOGW (tag, format, ...)

Log at WARN level, gated by CONFIG_AT_LOG_DEFAULT_LEVEL.

ESP_AT_LOGI (tag, format, ...)

Log at INFO level, gated by CONFIG_AT_LOG_DEFAULT_LEVEL.

ESP_AT_LOGD (tag, format, ...)

Log at DEBUG level, gated by CONFIG_AT_LOG_DEFAULT_LEVEL.

ESP_AT_LOGV (tag, format, ...)

Log at VERBOSE level, gated by CONFIG_AT_LOG_DEFAULT_LEVEL.

6.21 如何配置静默模式

6.21.1 简介

静默模式 (silence mode) 是 ESP-AT 固件的一种编译配置选项，用于控制 AT 的日志（即 [AT 日志端口](#) 输出的日志）行为，通常建议禁用。如果启用，将减少应用固件的大小。但建议你 **优先考虑** 下面两种方式减少应用固件的大小：

- 在 `python build.py menuconfig` > Component config > AT 路径下禁用不需要的 AT 功能。
- 参考 [最小化二进制文件大小](#) 文档中的优化方法。

表 12: 启用和禁用静默模式的优点和缺点对比

静默模式	优点	缺点
禁用静默模式（即设置为 0），推荐此配置	开启一些 AT 的日志	应用固件 (esp-at.bin) 的大小会增加，这可能会导致编译 AT 固件时，应用固件大小超过了代码中划分的应用分区大小，从而导致编译失败（ 编译失败日志 ）。
启用静默模式（即设置为 1），不推荐此配置	应用固件 (esp-at.bin) 的大小会减小	此模式会删除一些 AT 的日志，并会使 CONFIG_LOG_DEFAULT_LEVEL 配置失效，从而无法输出相应等级日志，这会增加你调试 AT 功能的难度。

编译失败日志示例：

```
$ ./build.py build

... (more lines of build ESP-AT)

Generated /path/to/esp-at/build/esp-at.bin
[836/838] cd /path/to/esp-at/build/esp-at.bin
FAILED: esp-idf/esptool_py/CMakeFiles/app_check_size
cd /path/to/esp-at/build/esp-idf/esptool_py && [...] /bin/python /path/to/esp-at/
↳ esp-idf/components/partition_table/check_sizes.py --offset 0x8000 partition --
↳ type app /path/to/esp-at/build/partition_table/partition-table.bin /path/to/esp-
↳ at/build/esp-at.bin
Error: app partition is too small for binary esp-at.bin size 0x135ae0:
- Part 'ota_0' 0/16 @ 0xd0000 size 0x130000 (overflow 0x5ae0)
ninja: build stopped: subcommand failed.
```

6.21.2 配置

- 启用或禁用静默模式配置：
 - [本地编译 ESP-AT 工程](#)
在第三步安装环境时配置 `silence mode` 为 No（或 Yes）来禁用（或启用）静默模式。
 - [网页编译 ESP-AT 工程](#)
在执行第 5.3 步提交更改时，修改 [esp-at/.github/workflows/build_template_esp32c6.yml](#) 文件中 `silence_mode` 的值，来启用或禁用静默模式。例如，要开启静默模式，则设置为：

```
- name: Configure prerequisites
  run: |
    # set module information
    silence_mode=1
    mkdir build
    echo -e "{\"platform\": \"PLATFORM_ESP32C6\", \"module\": \"${{ inputs.
↳ module_name }}\", \"silence\": ${silence_mode}\"" > build/module_info.json
```

- 请将编译好的 AT 固件烧录到模组上，然后执行 [AT+GMR](#) 命令，确认是否成功配置了静默模式的启用或禁用。
 - 如果启用静默模式，AT+GMR 命令响应中应包含 `s-`，例如：

```
AT+GMR
AT version:3.5.0.0-dev(s-88b4ea4...

... (more lines of version information)

OK
```

- 如果禁用静默模式，AT+GMR 命令响应中不包含 `s-`，例如：

```
AT+GMR
AT version:3.5.0.0-dev(88b4ea4...

... (more lines of version information)

OK
```

6.22 如何启用更多 AT 调试日志

本文主要介绍了如何启用日志配置选项。为了获取更多用于调试 ESP-AT 功能的日志信息，请自行[编译 ESP-AT 工程](#)，在第三步配置工程里选择 `silence mode` 为 `No`。在第五步配置工程里，根据你的实际情况，针对性地启用日志配置选项。

- [AT Wi-Fi 功能调试](#)
- [AT 网络功能调试](#)
- [AT BLE 功能调试](#)
- [AT 接口功能调试](#)
- [调试示例](#)

最终的调试日志，将从 [AT 日志端口](#) 输出，请根据[硬件连接](#)文档，查找 **输出日志**对应的 TX GPIO 管脚，通过串口工具查看该管脚的日志。

备注：启用日志配置选项后，日志输出量将显著增加，因此建议先确认问题属于哪个模块，再启用对应模块的日志配置。防止因日志输出量过大，导致无法复现或者出现 `task wdt` 等新的问题。

启用日志配置选项后，固件大小将会增加，可能导致编译失败。此时，你可以在 `python build.py menuconfig > Component config > AT` 路径下禁用不需要的 AT 功能来减小固件大小。

6.22.1 AT Wi-Fi 功能调试

- 你可以启用以下配置以开启 Wi-Fi 收发包的统计数据日志：

```
Component config > Log > Log Level > Default log verbosity > Info (或 Info_
→以上等级)
Component config > AT > Enable ESP-AT Debug
Component config > AT > Enable ESP-AT Debug > Periodically dump Wi-Fi_
→statistics
Component config > AT > Enable ESP-AT Debug > Periodically dump Wi-Fi_
→statistics > The interval of dumping Wi-Fi statistics (ms)
```

- 你可以启用以下配置以开启 Wi-Fi 交互过程的调试日志：

```
Component config > Log > Log Level > Default log verbosity > Debug
Component config > Wi-Fi > Print debug messages from WPA Supplicant
```

备注：在分析日志时，你可能会遇到 Wi-Fi 原因码，具体含义请参考 [Wi-Fi 原因代码](#)。

6.22.2 AT 网络功能调试

- [TCP 调试](#)
- [UDP 调试](#)
- [SSL 调试](#)
- [ICMP 调试](#)
- [lwIP 网络调试](#)

TCP 调试

若要监控 TCP 或基于 TCP 的协议（如 SSL、HTTP、MQTT、WebSocket 等）的传出 (TX) 和传入 (RX) 数据包，请启用以下配置，最终的日志中将打印 TCP 数据包信息，包括 IP 总长度、TCP 数据长度、TCP 序列号、TCP 确认号、TCP 源端口、目标端口和 TCP 标志位 (TCP flags)。其中，TCP 标志位是 CWR(128)、ECN(64)、Urgent(32)、Ack(16)、Push(8)、Reset(4)、SYN(2) 和 FIN(1) 的累积值。

```
Component config > Log > Log Level > Default log verbosity > Info (或
↪Info 以上等级)
Component config > AT > Enable ESP-AT Debug
Component config > AT > Enable ESP-AT Debug > Enable Network Debug
Component config > AT > Enable ESP-AT Debug > Enable Network Debug >
↪Enable the TCP packet debug messages
Component config > AT > Enable ESP-AT Debug > Enable Network Debug >
↪Enable the TCP packet debug messages > Specify the list of TCP port
↪numbers to monitor > 0
```

UDP 调试

若要监控 UDP 或基于 UDP 的协议（如 DHCP、DNS、SNTP、mDNS 等）的传出 (TX) 和传入 (RX) 数据包，请启用以下配置，最终的日志中将会打印 UDP 数据包信息，其中包括 IP 总长度、源端口、目标端口和 UDP 数据长度。

```
Component config > Log > Log Level > Default log verbosity > Info (或
↪Info 以上等级)
Component config > AT > Enable ESP-AT Debug
Component config > AT > Enable ESP-AT Debug > Enable Network Debug
Component config > AT > Enable ESP-AT Debug > Enable Network Debug >
↪Enable the UDP packet debug messages
Component config > AT > Enable ESP-AT Debug > Enable Network Debug >
↪Enable the UDP packet debug messages > Specify the list of outgoing UDP
↪(UDP TX) port numbers to monitor
Component config > AT > Enable ESP-AT Debug > Enable Network Debug >
↪Enable the UDP packet debug messages > Specify the list of incoming UDP
↪(UDP RX) port numbers to monitor
```

SSL 调试

你可以启用以下配置以开启 SSL 功能的调试日志：

```
Component config > Log > Log Level > Default log verbosity > Verbose
Component config > mbedTLS > Enable mbedTLS debugging
Component config > mbedTLS > Enable mbedTLS debugging > Set mbedTLS
↪debugging level > Verbose
```

ICMP 调试

你可以启用以下配置以开启 ICMP 功能 ([AT+PING](#)) 的调试日志：

```

Component config > Log > Log Level > Default log verbosity > Info (或 Info 以上等级)
Component config > AT > Enable ESP-AT Debug
Component config > AT > Enable ESP-AT Debug > Enable Network Debug
Component config > AT > Enable ESP-AT Debug > Enable Network Debug > Enable the ICMP packet debug messages

```

lwIP 网络调试

如果启用以上配置输出的日志仍无法满足你的调试需求,你可以在路径 Component config > LWIP > Enable LWIP Debug 下启用所需的调试配置。

6.22.3 AT BLE 功能调试

- 你可以直接在 **AT 日志端口** 查看断开原因码,从而调试 BLE 断开过程。
- 你可以启用以下配置以开启 BLE 功能的调试日志:

```

Component config > Log > Log Level > Default log verbosity > Debug
Component config > Bluetooth > NimBLE Options > NimBLE Host log verbosity > Debug logs

```

6.22.4 AT 接口功能调试

- 当你想要在 **AT 日志端口** 获取到 AT 通过 TX 发送给 MCU 的数据时,你可开启如下配置:

```

Component config > Log > Log Level > Default log verbosity > Info (或 Info 以上等级)
Component config > AT > Enable ESP-AT Debug
Component config > AT > Enable ESP-AT Debug > Logging the data sent from AT to MCU (AT ---> MCU)
Component config > AT > Enable ESP-AT Debug > Logging the data sent from AT to MCU (AT ---> MCU) > The maximum length of the data sent from AT to MCU to be logged > 8192

```

- 当你想要在 **AT 日志端口** 获取 AT 通过 RX 接收到的 MCU 发送数据时,你可开启如下配置:

```

Component config > Log > Log Level > Default log verbosity > Info (或 Info 以上等级)
Component config > AT > Enable ESP-AT Debug
Component config > AT > Enable ESP-AT Debug > Logging the data received by AT from MCU (AT <---- MCU)
Component config > AT > Enable ESP-AT Debug > Logging the data received by AT from MCU (AT <---- MCU) > The maximum length of the data received by AT from MCU to be logged > 8192

```

6.22.5 调试示例

示例 1: 调试 TCP 连接的数据发送和接收过程

场景: ESP 模组在波特率为 115200 以及使用双向流控情况下,作为 TCP 客户端与服务器建立了 TCP 连接,进入透传模式,并发送数据给服务器。

如图中箭头所示:

- ESP-AT 发送 (TX) 的数据流为 S1 -> S2 -> S3 -> S4 -> S5 -> S6 -> S7 -> S8
- ESP-AT 接收 (RX) 的数据流为 R8 -> R7 -> R6 -> R5 -> R4 -> R3 -> R2 -> R1

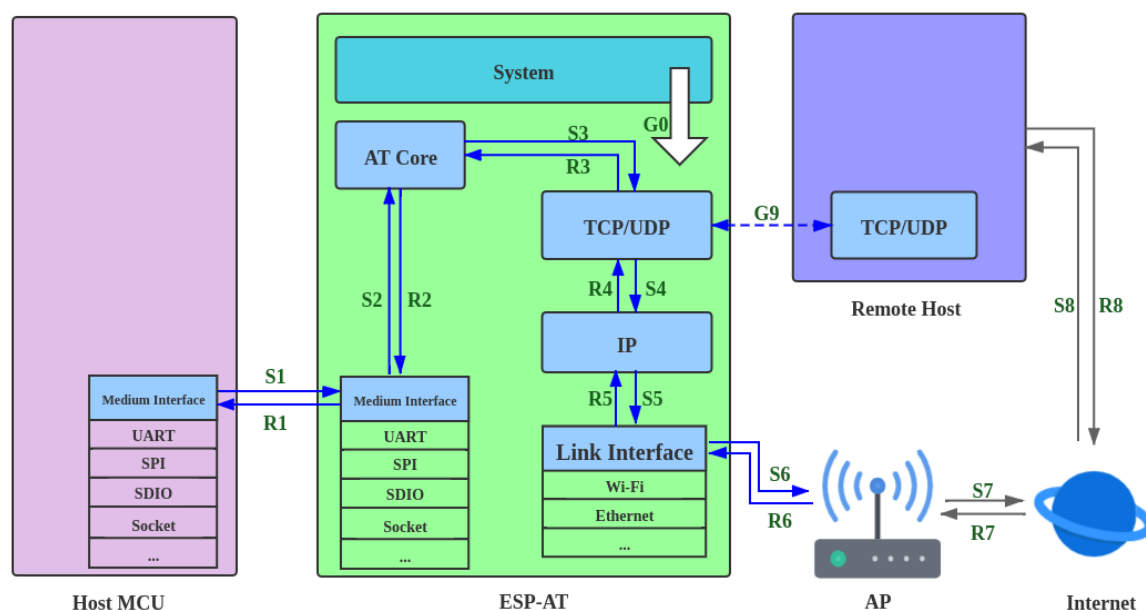


图 38: 数据流

如需排查 AT 固件在各通信层级（串口通信、AT 接口、网络、Wi-Fi 等）中的数据传输问题，请参考以下分层指导，选择相应的调试方式进行分析：

- 对于 S1/R1（串口通信）：
使用逻辑分析仪捕获 MCU 向 ESP 模组发送的串口数据（TX -> RX 线路），以确认 UART 通信是否正常。如果发现异常（通常表现为数据错误和数据丢失），请先检查线路。常见的原因包括波特率过高、MCU 和 ESP 的 UART 线路太长、双向流控配置错误、硬件设计不合理等。
- 对于 S2/R2（AT 接口层）：
启用 [AT 接口调试](#) 配置并重新编译 AT 固件。复测场景时，通过 [AT 日志端口](#) 输出的日志检查 RX（接收）和 TX（发送）数据是否正常。如果发现数据丢失，请先确认硬件流控是否开启。否则如果确认为 AT 引发的问题，请在 [esp-at/issues](#) 提交 issue。
- 对于 S3/R3（AT 内部处理）：
无需用户关注。
- 对于 S4-S5/R4-R5（网络层）：
启用 [TCP 调试](#) 配置并重新编译 AT 固件。复测场景时，通过 [AT 日志端口](#) 输出的日志分析 LwIP 层的传输是否正常。如果发现问题，请在 [esp-at/issues](#) 提交 issue。
- 对于 S6/R6（Wi-Fi 层）：
启用 [Wi-Fi 调试](#) 配置并重新编译 AT 固件。复测场景时，参考 [乐鑫 Wireshark 使用指南](#) 捕获空中数据包，并结合 [AT 日志端口](#) 的日志分析 Wi-Fi 层数据传输是否正常。如果确认问题出在 Wi-Fi 层，请在 [esp-at/issues](#) 提交 issue；若问题源自服务器端，请自行排查。

示例 2：调试 BLE 连接、扫描过程

场景：ESP 模组作为 BLE 客户端与手机反复建立 BLE 连接和断开连接。

- 若你想调试 BLE 断开的原因，请在复测场景时，抓取 [AT 日志端口](#) 的调试日志进行分析。日志中可能包含断开原因码，这些原因码是 BLE 通用的，并非 AT 特有。示例：

```
Disconnect reason = 0x13
```

- 若你想调试 BLE 扫描或连接等过程，请启用 [AT BLE 功能](#) 调试配置，并编译 AT 固件。在复测场景时，抓取 [AT 日志端口](#) 调试日志进行分析。

Chapter 7

第三方开放云平台定制化 AT 命令和固件

待开发。

如果你对第三方开放云平台的定制 AT 命令和固件有任何需求，请联系 [乐鑫商务组](#) 进行评估。

Chapter 8

AT FAQ

- AT 文档
 - 如何选择正确的 AT 文档版本？
- AT 固件
 - 我的模组没有官方发布的固件，如何获取适用的固件？
 - 如何获取 AT 固件源码？
 - 官网上放置的 AT 固件如何下载？
 - 如何整合 ESP-AT 编译出来的所有 bin 文件？
 - 新购买的 ESP 系列模组上电后，串口打印错误 “flash read err” 是什么原因？如何使用 AT 命令？
 - 模组出厂 AT 固件是否支持流控？
- AT 命令与响应
 - AT 提示 busy p... 是什么原因？
 - 上电后发送第一个命令总是返回错误信息？
 - 在不同模组上的默认 AT 固件支持哪些命令？
 - MCU 给设备发 AT 命令无返回，是什么原因？
 - AT 命令中特殊字符如何处理？
 - AT 命令中串口波特率是否可以修改？
 - 如何修改 AT 命令端口的 UART 管脚？
 - 如何解决 TCP/SSL 数据传输时的丢失问题？
 - 透传模式热点断开能否给出提示信息？
 - Wi-Fi 断开是什么原因？
 - Wi-Fi 常见的兼容性问题有哪些？
 - AT 命令是否支持 ESP-WIFI-MESH？
 - 是否有 AT 命令连接阿里云以及腾讯云示例？
 - AT 命令是否可以设置低功耗蓝牙发射功率？
 - 低功耗蓝牙客户端如何使能 notify 和 indicate 功能？
- 硬件
 - 在不同模组上的 AT 固件要求芯片 flash 多大？
 - AT 固件如何查看 error log？
 - AT 在模组上的 UART1 通信管脚不一致？
- 性能
 - AT Wi-Fi 连接耗时多少？
 - AT 固件中 TCP 发送窗口大小是否可以修改？
 - AT 吞吐量如何测试及优化？
 - 如何修改 AT 默认 TCP 数据段最大重传次数？

- 其他

- 乐鑫芯片可以通过哪些接口来传输 AT 命令？
- AT 如何使能调试日志？
- 如何修改 AT 固件首次启动时的默认 Wi-Fi 模式？
- AT 指令如何实现 HTTP 断点续传功能？
- 如何从 HTTP 服务器下载文件并存储到 FATFS，或将 FATFS 文件系统中的文件上传到 HTTP 服务器？
- 使用 *nmap* 等工具扫描发现 AT 设备存在开放端口，是否说明固件存在安全风险？
- 使用 *esp-idf-sbom* 扫描 AT 固件出现大量 CVE，是否会影响 RED、EN 18031、CRA 等合规？

8.1 AT 文档

8.1.1 如何选择正确的 AT 文档版本？

请根据实际使用的 AT 固件版本，选择对应芯片平台和文档版本；不同固件版本之间可能存在功能和行为差异。

- 对于官方发布版固件，可直接通过 `AT+USERDOCS?` 命令获取对应文档链接，无需额外确认版本信息。
- 对于非官方发布版固件，请先确认以下信息，再选择与之相匹配的文档版本：
 - 所用 AT 固件对应的芯片平台（如 ESP32、ESP32-C3 等）
 - AT 固件版本号（可通过 `AT+GMR` 命令查看 AT 核心库的版本号）



8.2 AT 固件

8.2.1 我的模组没有官方发布的固件，如何获取适用的固件？

如果 AT 固件 中没有相关固件，可：

- 使用相同硬件配置的模组固件（参考 [ESP-AT 固件差异](#)）
- 参考：[我该选哪种类型的 AT 固件？](#)

8.2.2 如何获取 AT 固件源码？

esp-at 项目以源代码和预编译库的组合形式在此仓库中分发。预编译的核心库（位于 `esp-at/components/at/lib/` 目录下）是闭源的，无开源计划。

8.2.3 官网上放置的 AT 固件如何下载？

使用 [Flash 下载工具](#) 烧录，详细指导参阅 [Flash 下载工具用户指南](#)。烧录地址参考 [AT 下载指南](#)。

8.2.4 如何整合 ESP-AT 编译出来的所有 bin 文件？

无需手动整合。发布的固件和编译生成的固件已自带合成好的 2MB/4MB bin 文件，位于 build/factory 目录下。

8.2.5 新购买的 ESP 系列模组上电后，串口打印错误 “flash read err” 是什么原因？该模组如何使用 AT 命令？

ESP 系列模组出厂时 Flash 中未烧录 ESP-AT 固件，上电后会提示 “flash read err” 错误。如需使用 AT 命令功能，请参考 [AT 下载指南](#)。

8.2.6 模组出厂 AT 固件是否支持流控？

支持硬件流控，不支持软件流控。可通过 [AT+UART_CUR](#) 或 [AT+UART_DEF](#) 配置硬件流控。硬件接线参考 [硬件连接](#)。

8.3 AT 命令与响应

8.3.1 AT 提示 busy p... 是什么原因？

提示 “busy p...” 表示正在处理前一条命令，无法响应当前输入。AT 命令处理是线性的，必须等前一条命令完成后才能接收下一条。

当有多余的不可见字符输入时（如命令后的空格），系统也会提示 “busy p...” 或 “ERROR”，因为任何串口输入均被视为命令输入。

如果 MCU 发送 AT+CIPSEND 后收到 busy p... 响应，表示上一条命令正在执行，当前输入无效。建议等待上一条命令响应完成后再发送新命令。

8.3.2 AT 固件，上电后发送第一个命令总是会返回下面的信息，为什么？

```
ERR CODE:0x010b0000
busy p...
```

表示“正在处理上一条命令”。一般只显示 “busy p...”，显示 ERR CODE 是因为打开了错误代码提示。

如果上电后的第一条命令就返回此错误，可能原因：

- 命令后多跟了换行符/空格/其他符号
- 连续发送了多个 AT 命令

8.3.3 在不同模组上的默认 AT 固件支持哪些命令，以及哪些命令从哪个版本开始支持？

各模组默认固件支持的命令：参考 [ESP-AT 固件差异](#)

命令版本支持和修复问题：参考 [release notes](#)

8.3.4 MCU 给 ESP32-C6 设备发 AT 命令无返回，是什么原因？

发送 AT 命令后需添加结束符号，程序中写法为：“AT\r\n”。参考[检查 AT 固件是否烧录成功](#)。

8.3.5 AT 命令中特殊字符如何处理？

可以参考[AT 命令分类](#) 章节中的转义字符语法。

8.3.6 AT 命令中串口波特率是否可以修改？(默认：115200)

可以修改，方法：

- 第一种方法，你可以通过串口命令 `AT+UART_CUR` 或 `AT+UART_DEF`。
- 第二种方法，你可以重新编译 AT 固件，编译介绍：[如何编译 AT 工程与修改 UART 波特率配置](#)。
- 第三种方法，你可以使用 `AT+SYSMFG` 命令修改出厂默认的 UART 波特率，请参考[AT+SYSMFG 命令示例](#)。

8.3.7 如何修改 AT 命令端口的 UART 管脚？

有多种方法可以修改 AT 命令端口的 UART 管脚：

- 第一种方法，请参考[如何设置 AT 端口管脚](#) 文档，修改 `factory_param_data.csv` 文件中的 UART 配置参数，然后重新编译和烧录固件。
- 第二种方法，如果你想在运行时通过 AT 命令修改 UART 配置而无需重新编译固件，请参考[AT+SYSMFG 命令示例](#)。此方法可以动态修改出厂默认的 UART 配置，包括波特率、管脚等参数。
- 第三种方法，如果你已有 AT 固件并且只需要修改 UART 管脚，可以使用 `at.py` 工具直接修改固件参数，无需重新编译。

8.3.8 如何解决 TCP/SSL 数据传输时的丢失问题？

若在未启用硬件流控时出现数据丢失，建议先启用硬件流控（`AT+UART_CUR` 或 `AT+UART_DEF`）验证是否由此导致。若启用流控后问题仍存在，请参考[调试 TCP 连接的数据发送和接收过程](#) 进一步排查；如确认为 AT 引发的问题，请在 [esp-at/issues](#) 提交 issue。

8.3.9 ESP32-C6 使用 AT 命令进入透传模式，如果连接的热点断开，ESP32-C6 能否给出相应的提示信息？

可通过 `AT+SYSMSG` 配置，设置 `AT+SYSMSG=4`，热点断开时串口会上报 `WIFI_DISCONNECT\r\n`。

注意：该命令在 AT v2.1.0.0 之后添加，v2.1.0.0 及之前的版本无法使用。

8.3.10 Wi-Fi 为什么断开 (WIFI DISCONNECT)？

可在 [AT 日志端口](#) 查看断开原因，通常打印 `wifi disconnected, rc:<reason_code>`。原因代码说明见 [Wi-Fi Reason Code](#)。

通用固件可能无法提供详细原因。若需获取具体信息，需在代码中注册 Wi-Fi disconnect 事件并输出原因代码。

8.3.11 Wi-Fi 常见的兼容性问题有哪些？

- AMPDU 兼容性问题。
 - 如果路由器不支持 AMPDU，那么 ESP32-C6 会在和路由器交互时，自动关闭 AMPDU 功能。
 - 如果路由器支持 AMPDU，但是路由器和 ESP32-C6 之间的 AMPDU 传输存在兼容性问题，那么建议关闭路由器的 AMPDU 功能或者 ESP32-C6 的 AMPDU 功能。如果你要禁用 ESP32-C6 的 AMPDU 功能，请自行[编译 ESP-AT 工程](#)，在第五步配置工程里选择：
 - * 禁用 Component config>Wi-Fi>WiFi AMPDU TX
 - * 禁用 Component config>Wi-Fi>WiFi AMPDU RX
- phy mode 兼容性问题。如果路由器和 ESP32-C6 之间的 phy mode 存在兼容性问题，那么建议切换路由器的 phy mode 或者 ESP32-C6 的 phy mode。如果你要切换 ESP32-C6 的 phy mode，请参考[AT+CWSTAPROTO](#) 命令。

8.3.12 AT 命令是否支持 ESP-WIFI-MESH？

AT 当前不支持 ESP-WIFI-MESH。

8.3.13 是否有 AT 命令连接阿里云以及腾讯云示例？

- 阿里云示例：[AT+MQTT aliyun](#)
- 腾讯云示例：[AT+MQTT QCloud](#)

8.3.14 AT 命令是否可以设置低功耗蓝牙发射功率？

可以。ESP32-C6 的 Wi-Fi 和 Bluetooth LE 共用一根天线，使用[AT+RFPOWER](#) 命令设置。

8.3.15 低功耗蓝牙客户端如何使能 notify 和 indicate 功能？

- 低功耗蓝牙的特征的属性除了读、写还有 notify 和 indicate。这两种都是服务端向客户端发送数据的方式，但是要想真的发送成功需要客户端提前注册 notification，也就是写 CCCD 的值。
- 如果要使能 notify，需要写 0x01；如果要使能 indicate，需要写 0x02（写 0x2902 这个描述符）；如果是既想使能 notify 又想使能 indicate，需要写 0x03。
- 比如，ESP-AT 的默认的服务中，0xC305 是可 notify 的，0xC306 是可 indicate 的。我们分别写这两个特征下面的 0x2902 描述符：

```
AT+BLEGATTCWR=0,3,6,1,2
>
// 写低位 0x01 高位 0x00 (如果要使用 hex 格式写的话就是: 0100)
OK
// server: +WRITE:0,1,6,1,2,<0x01>,<0x00>
AT+BLEGATTCWR=0,3,7,1,2
>
// 写低位 0x02 高位 0x00 (如果要使用 hex 格式写的话就是: 0200)
OK
// server: +WRITE:0,1,6,1,2,<0x02>,<0x00>
// 写 ccc 是 server 可以发送 notify 和 indicate 的前提条件
```

8.4 硬件

8.4.1 在不同模组上的 AT 固件要求芯片 flash 多大？

参考[ESP-AT 固件差异](#)。

8.4.2 AT 固件如何查看 error log ?

通常从 [AT 日志端口](#) 查看 error log。

8.4.3 AT 在 ESP32-C6 模组上的 UART1 通信管脚与 ESP32-C6 模组的 datasheet 默认 UART1 管脚不一致 ?

ESP32-C6 支持 IO 矩阵变换, 编译 ESP-AT 时可在 menuconfig 中配置 UART1 管脚, 因此可能与 datasheet 默认管脚不一致。管脚详情参考 [factory_param_data.csv](#)。

8.5 性能

8.5.1 AT Wi-Fi 连接耗时多少 ?

办公室场景下实测约 5 秒。实际耗时取决于路由器性能、网络环境、模块天线性能等因素。可通过 [AT+CWJAP](#) 的 `<jap_timeout>` 参数设置最大超时时间。

8.5.2 AT 固件中 TCP 发送窗口大小是否可以修改 ?

无法通过命令修改, 需重新编译固件。在 menuconfig 中配置: Component config > LWIP > TCP > Default send buffer size。

8.5.3 AT 吞吐量如何测试及优化 ?

建议使用 esp-idf 中的 iperf 示例进行测试 (AT 测试时请使用透传方式, 并将数据量调整为 1460 字节连续发送)。若速率不满足需求, 参考 [如何提高 ESP-AT 吞吐性能](#) 进行优化。

8.5.4 如何修改 AT 默认 TCP 数据段最大重传次数 ?

默认重传最大次数为 6 次, 取值范围 [3-12]。重新配置方法:

- 本地编译: 参考 [本地编译 ESP-AT 工程](#), 在 menuconfig 中配置:

```
Component config > LWIP > TCP > Maximum number of retransmissions of data ↵
↵ segments
```

- 网页编译: 参考 [网页编译 ESP-AT 工程](#), 修改 CONFIG_LWIP_TCP_MAXRTX 的值。

8.6 其他

8.6.1 乐鑫芯片可以通过哪些接口来传输 AT 命令 ?

- ESP32-C6 支持 UART、SPI 接口通信
- AT 默认固件使用 UART 接口。如需使用 SDIO 或 SPI 接口, 需基于 ESP-AT 配置编译, 详情见 [编译和开发](#)
- 参考文档: [使用 AT SDIO 接口](#), [使用 AT SPI 接口](#), [使用 AT 套接字接口](#)

8.6.2 AT 如何使能调试日志 ?

参考 [如何启用更多 AT 调试日志](#) 文档。

8.6.3 如何修改 AT 固件首次启动时的默认 Wi-Fi 模式？

修改源码，在 `esp_at_ready_before()` 函数中调用 `esp_wifi_set_mode()` 设置 Wi-Fi 模式，例如 `esp_wifi_set_mode(WIFI_MODE_STA)`。

8.6.4 AT 指令如何实现 HTTP 断点续传功能？

两种方法：

- 使用 HTTP 的 Range 字段指定读取范围，参考 [AT+HTTPCHEAD 示例](#)
- 使用 AT TCP 指令自行构造 HTTP GET 请求。在 [ESP32-C6 设备获取被动接收模式下的套接字数据示例](#) 的第 6 步和第 7 步之间，使用 [AT+CIPSEND](#) 发送构造的 HTTP GET 请求。被动接收模式下，使用 [AT+CIPRECVDATA](#) 读取数据

8.6.5 如何从 HTTP 服务器下载文件并存储到 FATFS，或将 FATFS 文件系统中的文件上传到 HTTP 服务器？

参考 GitHub 示例：

- [at_http_get_to_fs](#) - 从 HTTP 服务器下载文件并存储到 FATFS 文件系统
- [at_fs_to_http_server](#) - 将 FATFS 文件系统中的文件上传到 HTTP 服务器

8.6.6 使用 nmap 等工具扫描发现 AT 设备存在开放端口，是否说明固件存在安全风险？

结论：默认 AT 固件连上 Wi-Fi 后 **不会主动监听**任何端口。未通过 AT 命令创建连接或服务时，nmap 报告的 `open/open|filtered` **不是**固件在提供服务。

仅在使用 [AT+CIPSTART](#)、[AT+CIPSERVER](#)、[AT+WEBSEVER](#)、[AT+MDNS](#) 等命令时才会占用端口。

误判原因

- **UDP 扫描不可靠：**nmap -sU 靠 ICMP port-unreachable 判断端口关闭；未收到回复即标记 `open|filtered`，**不等于**端口在监听。UDP 及 ICMP 回复 **会丢包**，全端口高速扫描（如 -p 1-65535 -T4）误报更多，**不能**据此认定设备开放了端口。
- **扫错目标或已有连接：**确认 IP 与 [AT+CIFSR](#) 一致；高位 UDP 端口来自 [AT+CIPSTART](#) 套接字，5353 仅 AT+MDNS 启用后存在。

排查

1. AT+CIFSR、AT+CIPSTATE？—核对 IP 与当前连接，无多余监听即正常。
2. TCP 扫描：sudo nmap -sS -p- <IP>；未建连时应全部 closed。
3. 需 UDP 扫描时用 -T2 降速，或抓包确认 port-unreachable 是否到达。
4. AT+RST 后仅 AT+CWJAP 连 Wi-Fi，再复扫。

仍异常请记录扫描命令、[AT+GMR](#) 版本及 AT 命令序列，至 [esp-at/issues](#) 反馈。

8.6.7 使用 esp-idf-sbom 扫描 AT 固件出现大量 CVE，是否会影响 RED、EN 18031、CRA 等合规？

[esp-idf-sbom](#) 等工具报告的 CVE 针对的是软件组件（库）的特定版本，并不代表最终编译出的 AT 固件一定受影响。多数 CVE 只有在相关功能或代码路径被启用时才会编入固件并具备触发条件；对于 AT 固件中未启用的功能，其源码不会进入最终二进制，对应漏洞也就没有触发条件。因此，这类 CVE 通常不影响 AT 固件满足欧盟无线电设备指令（RED，2014/53/EU）的网络安全要求（委托条例 (EU) 2022/30 及协调标准 EN 18031 系列）以及《网络弹性法案》（CRA）等合规要求，提供一份自我评估 (self-assessment) 说明即可。

扫描结果之所以会高估风险，主要原因如下：

- 工具按组件版本号匹配公开漏洞库（如 NVD），只要版本落在受影响范围内即报告，并不分析该漏洞在具体编译配置下是否真正可达。
- ESP-AT 通过 menuconfig (Kconfig) 裁剪功能，被关闭功能对应的源码不会编入固件，其携带的漏洞也不存在于最终二进制中。
- 即使代码被编入固件，漏洞往往仍需多个前提条件同时满足才能触发，任一条件不满足即不可利用。

以 CVE-2026-34873（TLS 1.3 会话恢复期间的客户端伪装）为例，该漏洞影响 Mbed TLS 3.5.0 至 3.6.5 及 4.0.0（会被 SBOM 报告）。根据 [官方安全公告](#)，仅当以服务端角色运行的 Mbed TLS 同时满足以下条件时才可能受影响：

- 同时支持 TLS 1.2 与 TLS 1.3；
- 配置为对客户端进行身份认证；
- 向已认证的客户端签发 TLS 1.3 会话票据 (session ticket) 用于后续会话恢复。

AT 固件虽然可通过 [AT+CIPSTART](#) 作为 SSL 客户端、通过 [AT+CIPSERVER](#) 作为 SSL 服务端，但其 Mbed TLS 默认未启用 TLS 1.3（需在 menuconfig 中手动开启，参见 [修改 TLS 协议版本](#)），因此既不会协商 TLS 1.3，也不会签发 TLS 1.3 会话票据，上述条件无从满足。所以，尽管 SBOM 会因 Mbed TLS 版本报告该 CVE，但在默认 AT 固件配置下，该漏洞无法被触发，因此对固件没有实际影响。

建议按以下步骤处理被报告的 CVE：

1. 查阅每条 CVE 的官方描述（如 NVD、组件安全公告），确认其触发所需的配置与运行条件。
2. 对照 AT 固件实际情况，检查相关功能是否启用、代码是否被编译、运行时是否具备触发场景。可参考 [本地编译 ESP-AT 工程](#) 确认编译配置。
3. 对每条 CVE 标注受影响 / 不受影响及判定依据，作为合规所需的自我评估材料。
4. 若确认某条 CVE 在你的配置下确实可达，可升级到包含修复的组件版本或关闭相关功能后重新编译固件，也欢迎通过 [esp-at/issues](#) 联系我们。我们持续跟踪上游组件的安全公告，在适当情况下更新相关依赖，并将适用的安全修复集成到后续发布的 AT 固件版本中。

Chapter 9

Index of Abbreviations

A2DP

Advanced Audio Distribution Profile
高级音频分发框架

ADC

Analog-to-Digital Converter
模拟数字转换器

ALPN

Application Layer Protocol Negotiation
应用层协议协商

AT

AT stands for “attention” .
AT 是 attention 的缩写。

AT command port

The port that is used to send AT commands and receive responses. More details are in the [AT port](#) introduction.

AT 命令端口

也称为 AT 命令口，用于发送 AT 命令和接收响应的端口。更多介绍请参考[AT 端口](#)。

AT log port

The port that is used to output log. More details are in the [AT port](#) introduction.

AT 日志端口

也称为 AT 日志口，用于输出 AT 日志的端口。更多介绍请参考[AT 端口](#)。

AT port

AT port is the general name of AT log port (that is used to output log) and AT command port (that is used to send AT commands and receive responses). Please refer to [硬件连接](#) for default AT port pins and [如何设置 AT 端口管脚](#) for how to customize them.

AT 端口

AT 端口是 AT 日志端口（用于输出日志）和 AT 命令端口（用于发送 AT 命令和接收响应）的总称。请参考[硬件连接](#) 了解默认的 AT 端口管脚，参考[如何设置 AT 端口管脚](#) 了解如何自定义 AT 端口管脚。

Bluetooth LE

Bluetooth Low Energy
低功耗蓝牙

BluFi

Wi-Fi network configuration function via Bluetooth channel
BluFi 是一款基于蓝牙通道的 Wi-Fi 网络配置功能

Command Mode

Default operating mode of AT. In the command mode, any character received by the AT command port will be treated as an AT command, and AT returns the command execution result to the AT command port. AT enters [Data Mode](#) from [Command Mode](#) in the following cases.

- After sending the [AT+CIPSEND](#) set command successfully and returns >.
- After sending the [AT+CIPSEND](#) execute command successfully and returns >.
- After sending the [AT+CIPSENDL](#) set command successfully and returns >.

- After sending the *AT+CIPSENDEX* set command successfully and returns >.
- After sending the *AT+SAVETRANSLINK* set command successfully and sending the *AT+RST* command and restart the module.

In the data mode, send the *+++* command, AT will exit from *Data Mode* and enter the *Command Mode*.

命令模式

AT 的默认工作模式。在命令模式下，AT 命令端口收到的任何字符都会被当作 AT 命令进行处理，同时 AT 会在命令端口回复命令执行结果。AT 在下列情况下，会从命令模式进入数据模式。

- 发送 *AT+CIPSEND* 设置命令成功，回复 > 之后
- 发送 *AT+CIPSEND* 执行命令成功，回复 > 之后
- 发送 *AT+CIPSENDL* 设置命令成功，回复 > 之后
- 发送 *AT+CIPSENDEX* 设置命令成功，回复 > 之后
- 发送 *AT+SAVETRANSLINK* 设置命令成功，再发送 (*AT+RST*) 命令，模组重启之后

在数据模式下，发送 *+++* 命令，会从数据模式退出，进入命令模式。

Data Mode

In the data mode, any character received by the AT command port will be treated as data (except for special *+++*) instead of the AT command, and these data will be sent to the opposite end without modification. AT enters *Data Mode* from *Command Mode* in the following cases.

- After sending the *AT+CIPSEND* set command successfully and returns >.
- After sending the *AT+CIPSEND* execute command successfully and returns >.
- After sending the *AT+CIPSENDL* set command successfully and returns >.
- After sending the *AT+CIPSENDEX* set command successfully and returns >.
- After sending the *AT+SAVETRANSLINK* set command successfully and sending the *AT+RST* command and restart the module.

In the data mode, send the *+++* command, AT will exit from *Data Mode* and enter the *Command Mode*.

数据模式

在数据模式下，AT 命令端口收到的任何字符都会被当作数据（除了特殊的 *+++*），而不是 AT 命令，这些数据会无修改的发往对端。AT 在下列情况下，会从命令模式进入数据模式。

- 发送 *AT+CIPSEND* 设置命令成功，回复 > 之后
- 发送 *AT+CIPSEND* 执行命令成功，回复 > 之后
- 发送 *AT+CIPSENDL* 设置命令成功，回复 > 之后
- 发送 *AT+CIPSENDEX* 设置命令成功，回复 > 之后
- 发送 *AT+SAVETRANSLINK* 设置命令成功，再发送 *AT+RST* 命令，模组重启之后

在数据模式下，发送 *+++* 命令，会从数据模式退出，进入命令模式。

DHCP

Dynamic Host Configuration Protocol
动态主机配置协议

DNS

Domain Name System
域名系统

DTIM

Delivery Traffic Indication Map
延迟传输指示映射

GATTC

Generic Attributes client
GATT 客户端

GATTS

Generic Attributes server
GATT 服务器

HID

Human Interface Device
人机接口设备

I2C

Inter-Integrated Circuit
集成电路总线

ICMP

- Internet Control Message Protocol
因特网控制报文协议
- LwIP**
A Lightweight TCP/IP stack
一个轻量级的 TCP/IP 协议栈
- LWT**
Last Will and Testament
遗嘱
- MAC**
Media Access Control
MAC 地址
- mDNS**
Multicast Domain Name System
多播 DNS
- manufacturing nvs**
Manufacturing Non-Volatile Storage. `manufacturing nvs` stores all certificates, private keys, GATTS data, module information, Wi-Fi configurations, UART configurations, etc. The default values of these configurations are defined in `raw_data`. These configurations are finally built into the `mfg_nvs.bin` file and downloaded to the flash at the address defined in `at_customize.csv`.
一个适用于量产的 NVS。manufacturing nvs 中存储了 AT 固件默认所用到的所有证书、私钥、GATTS 数据、模组信息、Wi-Fi 配置、UART 配置等。这些配置信息，默认值在 `raw_data` 里，最终生成了 `mfg_nvs.bin`，烧录到 `at_customize.csv` 中定义的位置。
- MSB**
Most Significant Bit
最高有效位
- MTU**
maximum transmission unit
最大传输单元
- NVS**
Non-Volatile Storage
非易失性存储器
- Normal Transmission Mode**
Default Transmission Mode
In normal transmission mode, users can send AT commands. For examples, users can send MCU data received by AT command port to the opposite end of transmission by *AT+CIPSEND*; and the data received from the opposite end of transmission will also be returned to MCU through AT command port with additional prompt: *+IPD*.
During a normal transmission, if the connection breaks, ESP32-C6 will give a prompt and will not attempt to reconnect.
More details are in *Transmission Mode Shift Diagram*.
- 普通传输模式**
默认传输模式
在普通传输模式下，用户可以发送 AT 命令。例如，用户可以通过 *AT+CIPSEND* 命令，发送 AT 命令口收到的 MCU 数据到传输对端。从传输对端收到的数据，会通过 AT 命令口返回给 MCU，同时会附带 *+IPD* 信息。
普通传输模式时，如果连接断开，ESP32-C6 不会重连，并提示连接断开。
更多介绍请参考 *Transmission Mode Shift Diagram*。
- OWE**
Opportunistic Wireless Encryption. OWE is a Wi-Fi standard which ensures that the communication between each pair of endpoints is protected from other endpoints.
More details are in *Wikipedia*.
机会性无线加密。OWE 是一种 Wi-Fi 标准，它确保每对端点之间的通信受到保护，不受其他端点的影响。
更多介绍请参考 *维基百科*。
- Passthrough Mode**
Also called as “Passthrough Sending & Receiving Mode” .
In passthrough mode, users cannot send AT commands except special *+++* command. All MCU data received by AT command port will be sent to the opposite end of transmission without any modification; and the data

received from the opposite end of transmission will also be returned to MCU through AT command port without any modification.

Passthrough mode can be classified into Network Passthrough Mode and Bluetooth Passthrough Mode according to the transmission method. Network Passthrough Mode can be further classified into TCP Passthrough Mode, UDP Passthrough Mode and SSL Passthrough Mode according to the supported connection methods. Each connection method supports both client and server roles.

During the Network passthrough transmission, if the connection breaks, ESP32-C6 (as client) will keep trying to reconnect until +++ is input to exit the passthrough transmission; ESP32-C6 (as server) will shutdown the old connection and listen for a new connection until +++ is input to exit the passthrough transmission.

More details are in [Transmission Mode Shift Diagram](#).

透传模式

也称为“透传发送接收模式”。

在透传模式下，除特殊命令+++外，用户无法发送其他 AT 命令。所有通过 AT 命令端口接收的 MCU 数据将被原样转发至传输对端；同时，从传输对端接收的数据也将通过 AT 命令端口无损回传至 MCU。

透传模式按照传输方式可分为 Network 透传模式和 Bluetooth 透传模式。Network 透传模式按照支持的连接方式可分为 TCP 透传模式、UDP 透传模式和 SSL 透传模式，每种连接方式均支持客户端和服务端两种角色。

Network 透传模式传输时，如果连接断开，ESP32-C6 作为客户端时，会不停地尝试重连，此时单独输入+++退出透传，则停止重连；ESP32-C6 作为服务器时，会关闭连接同时监听新的连接，此时单独输入+++退出透传。

更多介绍请参考[Transmission Mode Shift Diagram](#)。

Transmission Mode Shift Diagram

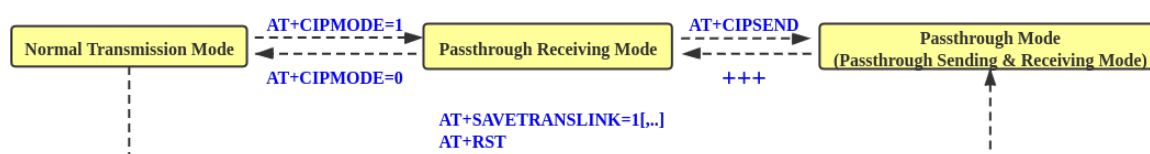


图 1: Transmission Mode Shift Diagram

More details are in the following introduction.

- [Normal Transmission Mode](#) (普通传输模式)
- [Passthrough Receiving Mode](#) (透传接收模式)
- [Passthrough Mode](#) (透传模式)
- [AT+CIPMODE](#)
- [AT+CIPSEND](#)
- [+++](#)
- [AT+SAVETRANSLINK](#)

Passthrough Receiving Mode

The temporary mode between [Normal Transmission Mode](#) and [Passthrough Mode](#).

In passthrough receiving mode, AT cannot send any data to the opposite end of transmission; but the data received from the opposite end of transmission can be returned to MCU through AT command port without any modification. More details are in [Transmission Mode Shift Diagram](#).

透传接收模式

在普通传输模式和透传模式之间的一个临时模式。

在透传接收模式，AT 不能发送数据到传输对端；但 AT 可以收到来自传输对端的数据，通过 AT 命令口无修改地返回给 MCU。更多介绍请参考[Transmission Mode Shift Diagram](#)。

PBC

Push Button Configuration
按钮配置

PCI Authentication

Payment Card Industry Authentication. In ESP-AT project, it refers to all Wi-Fi authentication modes except OPEN and WEP.

PCI 认证，在 ESP-AT 工程中指的是除 OPEN 和 WEP 以外的 Wi-Fi 认证模式。

PKI

A public key infrastructure (PKI) is a set of roles, policies, hardware, software and procedures needed to create, manage, distribute, use, store and revoke digital certificates and manage public-key encryption.

More details are in [Public Key Infrastructure](#).

公开密钥基础建设。公开密钥基础建设 (PKI) 是一组由硬件、软件、参与者、管理政策与流程组成的基础架构，其目的在于创造、管理、分配、使用、存储以及撤销数字证书。

更多介绍请参考 [公开密钥基础建设](#)。

PLCP

Physical Layer Convergence Procedure

PLCP 协议，即物理层会聚协议

PMF

protected management frame

受保护的管理帧

PSK

Pre-shared Key

预共享密钥

PWM

Pulse-Width Modulation

脉冲宽度调制

QoS

Quality of Service

服务质量

RTC

Real Time Controller. A group of circuits in SoC that keeps working in any chip mode and at any time.

实时控制器，为 SoC 中的一组电路，在任何芯片模式下都能随时保持工作。

SMP

Security Manager Protocol

安全管理协议

SNI

Server Name Indication

服务器名称指示

SNTP

Simple Network Time Protocol

简单网络时间协议

SPI

Serial Peripheral Interface

串行外设接口

SPP

Serial Port Profile

SPP 协议，即串口协议

SSL

Secure Sockets Layer

SSL 协议，即安全套接字协议

system message

Data sent via AT command port to MCU. Each system message usually ends with `\r\n`. Detailed system message descriptions are available at [AT Messages](#).

系统消息

AT 命令口发往 MCU 的数据。每条系统消息通常以 `\r\n` 结尾。详细的系统消息说明见 [AT 消息](#)。

TLS

Transport Layer Security

TLS 协议，即传输层安全性协议

URC

Unsolicited Result Code

非请求结果码，一般为模组给 MCU 的串口返回

UTC

Coordinated Universal Time

协调世界时

UUID

universally unique identifier

通用唯一识别码

WEP

Wired-Equivalent Privacy

WEP 加密方式，即有线等效加密

WPA

Wi-Fi Protected Access

Wi-Fi 保护访问

WPA2

Wi-Fi Protected Access II

Wi-Fi 保护访问 II

WAPI

WLAN Authentication and Privacy Infrastructure. WAPI is a wireless security protocol independently developed in China and designated as a national standard (GB 15629.11). It provides authentication and encryption mechanisms for wireless networks, using a different approach from WPA/WPA2.

More details are in [Wikipedia](#).

无线局域网鉴别和保密基础结构。WAPI 是中国自主研发的无线安全协议，是中国国家标准 (GB 15629.11)。它为无线网络提供身份验证和加密机制，其实现方式不同于 WPA/WPA2。

更多介绍请参考 [维基百科](#)。

WPS

Wi-Fi Protected Setup

Wi-Fi 保护设置

Chapter 10

特别声明

感谢你选择使用 ESP-AT 方案（固件）。为了确保你对本方案（本固件）的使用安全、合法且符合预期效果，请仔细阅读以下特别声明。本声明旨在明确用户在使用本方案（本固件）过程中需要遵循的各项条款和注意事项。通过继续使用本方案（本固件），你即表示接受并同意遵守这些条款。

10.1 应用场景验证

本方案（本固件）为通用版本，尽管针对各种场景做了大量的测试，但测试用例仍存在某些应用场景无法覆盖的风险，在决定基于本方案（本固件）进行商用量产前，请结合自己的应用场景，进行充足且全面的验证。

10.2 兼容性

由于技术日新月异，当某一新的技术、方案或标准出现时，可能导致兼容性问题，请结合自己产品做好相应的测试验证，并关注本方案（本固件）的新版本发布情况。

10.3 服务器升级注意事项

当对服务器进行升级时，包括但不限于操作系统升级、工具升级、证书升级等，请务必在正式升级前进行测试验证，避免因服务器变动导致的连接问题。

10.4 无线方案问题处理

无线方案易受外部因素影响，且问题定位复杂。请结合自身产品和云平台，规划好远程 OTA 和远程问题信息获取的解决方案，否则可能需要用户现场提供日志、数据包等必需的信息进行问题分析。

10.5 使用限制

用户不得将本方案（本固件）用于非法活动或违反相关法律法规的用途。用户应确保其使用本方案（本固件）的行为符合所有适用的法律和法规，我司不对因非法使用本方案（固件）而产生的任何后果负责。

10.6 责任限制

对于因使用本方案（本固件）造成的直接、间接、偶然、特殊或继发性的损害（包括但不限于数据丢失、利润损失、业务中断等），我司不承担任何责任，即使已被告知可能发生此类损害。

10.7 第三方软件和技术

本方案（本固件）可能包含第三方的软件或技术，我司对这些第三方软件或技术的适用性和合规性不承担责任。用户应遵循第三方的许可协议和使用条款。

10.8 数据保护

用户在使用本方案（本固件）过程中产生的数据应由用户自行负责保护和备份。我司对因数据丢失或泄露引起的任何损失不承担责任。

10.9 版本更新与维护

本方案（本固件）会不定期进行维护更新及版本发布，请自行关注版本状态，恕不另行通知。

10.10 声明更新

本声明条款可能会不定期更新，恕不另行通知。

10.11 用户责任

强烈建议用户认真阅读并严格遵守本声明条款，由于未遵守导致的风险，需由用户承担。

感谢你的理解与配合！

Chapter 11

关于 ESP-AT

这是 [ESP-AT](#) 的文档，ESP-AT 是乐鑫开发的可与乐鑫产品快速简单交互的解决方案。

乐鑫 Wi-Fi 和蓝牙芯片可以用作附加模块，可以完美集成在其他新产品或现有产品上，提供无线通讯功能。为降低客户开发成本，乐鑫开发了 [AT 固件](#) 和各类 [AT 命令](#)，方便客户简单快速地使用 AT 命令来控制芯片。



图 1: ESP-AT 方案

乐鑫提供的 AT 固件具有以下特色，利于芯片快速集成到应用中：

- 内置 TCP/IP 堆栈和数据缓冲
- 能便捷地集成到资源受限的主机平台中
- 主机对命令的回应易于解析
- 用户可自定义 AT 命令

11.1 其他 Espressif 连接方案

除了 ESP-AT 之外，乐鑫还提供了其他连接方案，以满足不同的集成需求：

- **ESP-Hosted 方案** 可使乐鑫系列芯片作为外部主机系统的无线通信协处理器。允许主机设备（如基于 Linux 的系统或微控制器 MCU）通过标准接口（如 SPI、SDIO 或 UART）添加 Wi-Fi 和 Bluetooth/BLE 功能。详情请见 [ESP-Hosted GitHub](#)。
- **ESP-IoT-Bridge 方案** 主要针对 IoT 应用场景下的各种网络接口之间的连接与通信，如 SPI、SDIO、USB、Wi-Fi、以太网等网络接口。在该解决方案中，Bridge 设备既可以为其它设备提供上网功能，又可以作为连接远程服务器的独立设备。详情请见 [ESP-IoT-Bridge GitHub](#)。

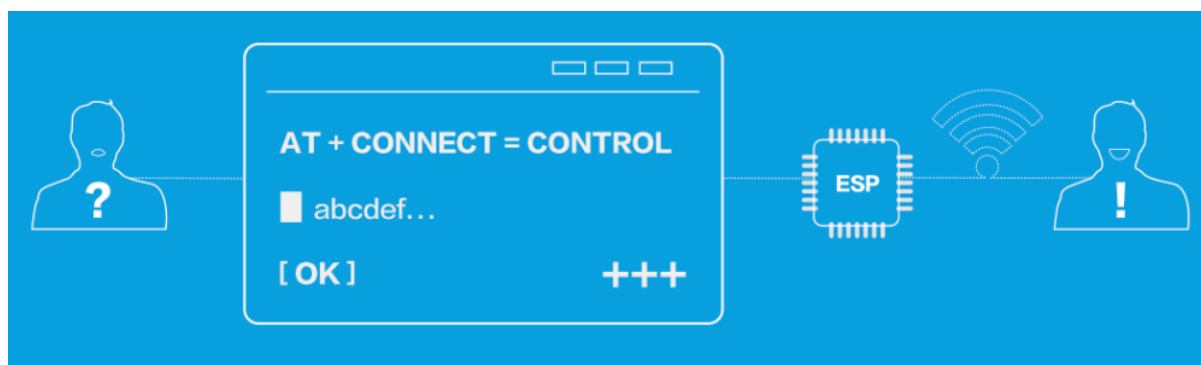


图 2: ESP-AT 命令

上述方案面向不同的系统架构和应用需求。对于希望在集成无线连接时寻找 AT 固件以外替代方案的用户，可以根据主机平台能力和系统设计目标选择更合适的解决方案。

- [genindex](#)

索引

A

A2DP, [445](#)
ADC, [445](#)
ALPN, [445](#)
AT, [445](#)
AT command port, [445](#)
AT log port, [445](#)
AT port, [445](#)
AT 命令端口, [445](#)
AT 日志端口, [445](#)
AT 端口, [445](#)

B

Bluetooth LE, [445](#)
BluFi, [445](#)

C

Command Mode, [445](#)
CONFIG_AT_LOG_DEFAULT_LEVEL (C macro), [429](#)

D

Data Mode, [446](#)
DHCP, [446](#)
DNS, [446](#)
DTIM, [446](#)

E

esp_at_ble_ops_t::connect_cb (C++ member), [422](#)
esp_at_ble_ops_t::disconnect_cb (C++ member), [422](#)
esp_at_ble_ops_t::recv_data (C++ member), [422](#)
esp_at_ble_ops_t (C++ struct), [421](#)
ESP_AT_BUF_ON_STACK_SIZE (C macro), [429](#)
ESP_AT_CMD_ERROR_CMD_EXEC_FAIL (C macro), [423](#)
ESP_AT_CMD_ERROR_CMD_OP_ERROR (C macro), [423](#)
ESP_AT_CMD_ERROR_CMD_PROCESSING (C macro), [423](#)
ESP_AT_CMD_ERROR_CMD_UNSUPPORTED (C macro), [423](#)
ESP_AT_CMD_ERROR_NON_FINISH (C macro), [422](#)

ESP_AT_CMD_ERROR_NOT_FOUND_AT (C macro), [423](#)
ESP_AT_CMD_ERROR_OK (C macro), [422](#)
ESP_AT_CMD_ERROR_PARA_INVALID (C macro), [423](#)
ESP_AT_CMD_ERROR_PARA_LENGTH (C macro), [423](#)
ESP_AT_CMD_ERROR_PARA_NUM (C macro), [423](#)
ESP_AT_CMD_ERROR_PARA_PARSE_FAIL (C macro), [423](#)
ESP_AT_CMD_ERROR_PARA_TYPE (C macro), [423](#)
esp_at_cmd_t::cmd_name (C++ member), [420](#)
esp_at_cmd_t::exe_cmd (C++ member), [421](#)
esp_at_cmd_t::query_cmd (C++ member), [420](#)
esp_at_cmd_t::setup_cmd (C++ member), [420](#)
esp_at_cmd_t::test_cmd (C++ member), [420](#)
esp_at_cmd_t (C++ struct), [420](#)
esp_at_custom_ble_ops_register (C++ function), [417](#)
esp_at_custom_cmd_array_register (C++ function), [416](#)
esp_at_custom_net_ops_register (C++ function), [416](#)
esp_at_custom_ops_register (C++ function), [417](#)
esp_at_custom_ops_t::pre_active_write_data_callback (C++ member), [422](#)
esp_at_custom_ops_t::pre_deepsleep_callback (C++ member), [422](#)
esp_at_custom_ops_t::pre_restart_callback (C++ member), [422](#)
esp_at_custom_ops_t::pre_sleep_callback (C++ member), [422](#)
esp_at_custom_ops_t::pre_wakeup_callback (C++ member), [422](#)
esp_at_custom_ops_t::status_callback (C++ member), [422](#)
esp_at_custom_ops_t (C++ struct), [422](#)
esp_at_custom_partition_find (C++ function), [417](#)
esp_at_device_ops_register (C++ function), [416](#)
esp_at_dispatch_result (C++ function), [419](#)

esp_at_errno_t::ESP_AT_SUB_CMD_EXEC_FAIL (C++ enumerator) , 426
 esp_at_errno_t::ESP_AT_SUB_CMD_OP_ERROR (C++ enumerator) , 426
 esp_at_errno_t::ESP_AT_SUB_CMD_PROCESSING (C++ enumerator) , 426
 esp_at_errno_t::ESP_AT_SUB_COMMON_ERROR (C++ enumerator) , 425
 esp_at_errno_t::ESP_AT_SUB_NO_AT (C++ enumerator) , 425
 esp_at_errno_t::ESP_AT_SUB_NO_TERMINATOR (C++ enumerator) , 425
 esp_at_errno_t::ESP_AT_SUB_OK (C++ enumerator) , 425
 esp_at_errno_t::ESP_AT_SUB_PARA_INVALID (C++ enumerator) , 425
 esp_at_errno_t::ESP_AT_SUB_PARA_LENGTH_ERROR (C++ enumerator) , 425
 esp_at_errno_t::ESP_AT_SUB_PARA_NUM_MISMATCH (C++ enumerator) , 425
 esp_at_errno_t::ESP_AT_SUB_PARA_PARSE_FAIL (C++ enumerator) , 425
 esp_at_errno_t::ESP_AT_SUB_PARA_TYPE_MISMATCH (C++ enumerator) , 425
 esp_at_errno_t::ESP_AT_SUB_UNSUPPORTED_CMD (C++ enumerator) , 426
 esp_at_errno_t (C++ enum) , 425
 ESP_AT_ERROR_NO (C macro) , 422
 esp_at_fs_get_mount_point (C++ function) , 428
 esp_at_fs_get_type (C++ function) , 428
 esp_at_fs_mount (C++ function) , 428
 esp_at_fs_unmount (C++ function) , 428
 esp_at_get_current_cmd_name (C++ function) , 417
 esp_at_get_current_module_name (C++ function) , 426
 esp_at_get_fs_info (C++ function) , 428
 esp_at_get_para_as_digit (C++ function) , 417
 esp_at_get_para_as_float (C++ function) , 417
 esp_at_get_para_as_str (C++ function) , 418
 esp_at_intf_ops_t::get_data_length (C++ member) , 421
 esp_at_intf_ops_t::read_data (C++ member) , 421
 esp_at_intf_ops_t::wait_write_complete (C++ member) , 421
 esp_at_intf_ops_t::write_data (C++ member) , 421
 esp_at_intf_ops_t (C++ struct) , 421
 ESP_AT_LOG_BUFFER_HEXDUMP (C macro) , 429
 esp_at_log_write (C++ function) , 426
 ESP_AT_LOGD (C macro) , 429
 ESP_AT_LOGE (C macro) , 429
 ESP_AT_LOGI (C macro) , 429
 ESP_AT_LOGV (C macro) , 429
 ESP_AT_LOGW (C macro) , 429
 esp_at_max (C macro) , 429
 esp_at_min (C macro) , 429
 esp_at_module_t::ESP_AT_MODULE_NUM (C++ enumerator) , 425
 esp_at_module_t (C++ enum) , 425
 esp_at_net_ops_t::connect_cb (C++ member) , 421
 esp_at_net_ops_t::disconnect_cb (C++ member) , 421
 esp_at_net_ops_t::recv_data (C++ member) , 421
 esp_at_net_ops_t (C++ struct) , 421
 esp_at_nvs_get_blob (C++ function) , 427
 esp_at_nvs_get_str (C++ function) , 427
 esp_at_nvs_set_blob (C++ function) , 427
 esp_at_nvs_set_str (C++ function) , 426
 esp_at_para_parse_ret_t::ESP_AT_PARA_PARSE_RET_FAIL (C++ enumerator) , 424
 esp_at_para_parse_ret_t::ESP_AT_PARA_PARSE_RET_OK (C++ enumerator) , 424
 esp_at_para_parse_ret_t::ESP_AT_PARA_PARSE_RET_OK_AND_MATCH (C++ enumerator) , 424
 esp_at_para_parse_ret_t (C++ enum) , 424
 esp_at_port_active_write_data_without_filter (C++ function) , 419
 esp_at_port_active_write_data (C++ function) , 418
 esp_at_port_enter_specific (C++ function) , 419
 esp_at_port_exit_specific (C++ function) , 419
 esp_at_port_get_data_length (C++ function) , 419
 esp_at_port_read_data (C++ function) , 418
 esp_at_port_recv_data_notify (C++ function) , 418
 esp_at_port_specific_callback_t (C++ type) , 423
 ESP_AT_PORT_TX_WAIT_MS_MAX (C macro) , 428
 esp_at_port_wait_write_complete (C++ function) , 419
 esp_at_port_write_data_without_filter (C++ function) , 418
 esp_at_port_write_data (C++ function) , 418
 esp_at_rc_t::ESP_AT_RESULT_CODE_ERROR (C++ enumerator) , 424
 esp_at_rc_t::ESP_AT_RESULT_CODE_FAIL (C++ enumerator) , 424
 esp_at_rc_t::ESP_AT_RESULT_CODE_IGNORE (C++ enumerator) , 425
 esp_at_rc_t::ESP_AT_RESULT_CODE_MAX (C++ enumerator) , 425
 esp_at_rc_t::ESP_AT_RESULT_CODE_OK_AND_INPUT_PROMPT (C++ enumerator) , 425

- (C++ enumerator) , 425
 - esp_at_rc_t::ESP_AT_RESULT_CODE_OK (C++ enumerator) , 424
 - esp_at_rc_t::ESP_AT_RESULT_CODE_PROCESS_DONE (C++ enumerator) , 425
 - esp_at_rc_t::ESP_AT_RESULT_CODE_SEND_FAIL (C++ enumerator) , 424
 - esp_at_rc_t::ESP_AT_RESULT_CODE_SEND_OK (C++ enumerator) , 424
 - esp_at_rc_t (C++ enum) , 424
 - esp_at_ready_before (C++ function) , 426
 - esp_at_restart_async (C++ function) , 420
 - esp_at_restart (C++ function) , 420
 - esp_at_sleep_mode_t::ESP_AT_SLEEP_DISABLE (C++ enumerator) , 424
 - esp_at_sleep_mode_t::ESP_AT_SLEEP_LIGHT (C++ enumerator) , 424
 - esp_at_sleep_mode_t::ESP_AT_SLEEP_MAX_MODEM (C++ enumerator) , 424
 - esp_at_sleep_mode_t::ESP_AT_SLEEP_MIN_MODEM (C++ enumerator) , 424
 - esp_at_sleep_mode_t::ESP_AT_SLEEP_MODE_MAX (C++ enumerator) , 424
 - esp_at_sleep_mode_t (C++ enum) , 423
 - esp_at_status_t::ESP_AT_STATUS_NORMAL (C++ enumerator) , 423
 - esp_at_status_t::ESP_AT_STATUS_TRANSMIT (C++ enumerator) , 423
 - esp_at_status_t (C++ enum) , 423
 - esp_at_str_2_mac (C++ function) , 420
 - esp_at_write_result (C++ function) , 419
 - esp_at_yield_if_idle_timeout (C++ function) , 428
- ## G
- GATTC, 446
 - GATTS, 446
- ## H
- HID, 446
- ## I
- I2C, 446
 - ICMP, 446
- ## L
- LwIP, 447
 - LWT, 447
- ## M
- MAC, 447
 - manufacturing nvs, 447
 - mDNS, 447
 - MSB, 447
 - MTU, 447
- ## N
- Normal Transmission Mode, 447
 - NVS, 447
- ## O
- ## P
- Passthrough Mode, 447
 - Passthrough Receiving Mode, 448
 - PBC, 448
 - PCI Authentication, 448
 - PKI, 448
 - PLCP, 449
 - PMF, 449
 - PSK, 449
 - PWM, 449
- ## Q
- ## R
- RTC, 449
- ## S
- SMP, 449
 - SNI, 449
 - SNTP, 449
 - SPI, 449
 - SPP, 449
 - SSL, 449
 - system message, 449
- ## T
- TLS, 449
 - Transmission Mode Shift Diagram, 448
- ## U
- URC, 449
 - UTC, 449
 - UUID, 449
- ## W
- WAPI, 450
 - WEP, 450
 - WPA, 450
 - WPA2, 450
 - WPS, 450
- ❓
- 命令模式, 446
- ❓
- 数据模式, 446
 - 普通传输模式, 447
- ❓
- 系统消息, 449



透传接收模式, [448](#)
透传模式, [448](#)